mathematics

MDPI

Article

The Improved Element-Free Galerkin Method for 3D Steady Convection-Diffusion-Reaction Problems with Variable Coefficients

Heng Cheng *, Zebin Xing and Yan Liu

School of Applied Science, Taiyuan University of Science and Technology, Taiyuan 030024, China
* Correspondence: chengheng@shu.edu.cn; Tel.: +86-178-3511-1258

Abstract: In order to obtain the numerical results of 3D convection-diffusion-reaction problems with variable coefficients efficiently, we select the improved element-free Galerkin (IEFG) method instead of the traditional element-free Galerkin (EFG) method by using the improved moving least-squares (MLS) approximation to obtain the shape function. For the governing equation of 3D convection-diffusion-reaction problems, we can derive the corresponding equivalent functional; then, the essential boundary conditions are imposed by applying the penalty method; thus, the equivalent integral weak form is obtained. By introducing the IMLS approximation, we can derive the final solved linear equations of the convection-diffusion-reaction problem. In numerical examples, the scale parameter and the penalty factor of the IEFG method for such problems are discussed, the convergence is proved numerically, and the calculation efficiency of the IEFG method are verified by four numerical examples.

Keywords: convection-diffusion-reaction; meshless method; improved element-free Galerkin method

MSC: 65N22

Citation: Cheng, H.; Xing, Z.; Liu, Y. The Improved Element-Free Galerkin Method for 3D Steady Convection-Diffusion-Reaction Problems with Variable Coefficients. *Mathematics* **2023**, *11*, 770. https://doi.org/10.3390/math11030770

Academic Editor: Rade Vignjevic

Received: 7 December 2022
Revised: 31 December 2022
Accepted: 4 January 2023
Published: 3 February 2023

1. Introduction

The convection-diffusion-reaction equation has been widely used in economics, chemistry, physics, and fluid mechanics fields. Because of the complexity of some problems with various coefficients, exact solutions are usually limited to only a few simplified cases. Therefore, how to obtain the numerical solutions of 3D convection-diffusion-reaction problems [1–4] with higher computational accuracy and computational speed is an important direction in the research of numerical methods.

As an important numerical method, the finite element method has been widely applied in many science and engineering fields, but the mesh distortion is not avoided when solving large deformation and crack propagation problems. As we know, meshless methods [5–10] are based on the scattered point approximation, which can avoid the mesh reconstruction, and, thus, a higher accuracy of the numerical solutions can be obtained.

In recent years, several meshless methods have been used to analyze convection-diffusion-reaction problems, such as the Galerkin and least squares method [11], variational multiscale element-free Galerkin methods [12–14], meshfree local Petrov Galerkin methods [15,16], Hermite method [17], and local knot method [18].

As an important meshless method, the element-free Galerkin (EFG) method [19] was invented by Belytschko et al., which uses the moving least-squares (MLS) approximation [20] to construct the shape function. This approximation is based on the ordinary least-squares method, which is the best approximation in mathematics [21], and it has been applied in engineering fields widely [22–24]. However, the singular function cannot be avoided and the calculation speed is slower. Afterward, the IMLS approximation [25] was proposed to improve the calculation speed of the MLS approximation by using the

orthogonal basis function. Thus, the IEFG method [26–30] was presented for some partial differential equations and mechanics problems, and the higher computational efficiency of the IEFG method was proved.

Generally, we use the penalty method or the Lagrange multiplier method to impose the essential boundary conditions when studying the EFG and the IEFG methods. In order to impose the essential boundary conditions directly, the interpolating MLS approximation based on the singular weight function was inverted [20]. Afterward, the improved interpolating MLS approximation [31] and the improved interpolating EFG method [32–35] were proposed. Qin et al. [36] studied the interpolating smoothed particle method.

Wang et al. [37] proposed another interpolating MLS method by employing the non-singular weight function instead of the singular weight function; thus, the corresponding interpolating EFG method [8] was presented by using this improved approximate function. Liu et al. used this interpolating EFG method to solve some large deformation problems [38–40].

Combining the complex theory with the MLS approximation, Cheng et al. [41] studied the complex variable moving least-squares (CVMLS) approximation. Based on the conjugate basis function, Bai et al. [42] proposed the improved complex variable moving least-squares (ICVMLS) approximation to obtain the shape functions, which can enhance the efficiency of the MLS approximation. Using the ICVMLS approximation to construct the shape function, the improved complex variable element-free Galerkin (ICVEFG) method was presented for elasticity [42] and wave propagation [43] problems. However, the ICVEFG method cannot be applied to 3D problems, due to the complication of the complex variable shape function.

In this paper, the IEFG method is selected to solve 3D convection-diffusion-reaction problems with variable coefficients. For the governing equation of such problems, we can derive the corresponding equivalent functional; then, the essential boundary conditions are imposed by applying the penalty method; thus, the equivalent integral weak form is obtained. By introducing the IMLS approximation, we can derive the final solved linear equations of convection-diffusion-reaction problems.

In numerical examples, the influence of the scale parameter, penalty factor, and the nodes distribution on numerical accuracy are discussed, the convergence is demonstrated numerically, and the correctness and the efficiency of the IEFG method are verified by four numerical examples. Additionally, it can avoid singular matrices that often exist in the EFG method.

2. The IMLS Approximation

The approximation of a function $u(\boldsymbol{x})$ is

$$u^h(\boldsymbol{x}) = \sum_{i=1}^{m} p_i(\boldsymbol{x}) \cdot a_i(\boldsymbol{x}) = \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{x}) \cdot \boldsymbol{a}(\boldsymbol{x}), \ (\boldsymbol{x} \in \Omega), \tag{1}$$

where $\boldsymbol{p}^{\mathrm{T}}(\boldsymbol{x})$ is the basis function vector, m is the basis function number, and

$$\boldsymbol{a}^{\mathrm{T}}(\boldsymbol{x}) = (a_1(\boldsymbol{x}), a_2(\boldsymbol{x}), \cdots, a_m(\boldsymbol{x})) \tag{2}$$

is the coefficient vector of $\boldsymbol{p}^{\mathrm{T}}(\boldsymbol{x})$.

In general,

$$\boldsymbol{p}^{\mathrm{T}}(\boldsymbol{x}) = (1, x_1, x_2, x_3), \tag{3}$$

$$\boldsymbol{p}^{\mathrm{T}}(\boldsymbol{x}) = (1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3, x_1x_3). \tag{4}$$

The local approximation is

$$u^h(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \sum_{i=1}^{m} p_i(\hat{\boldsymbol{x}}) \cdot a_i(\boldsymbol{x}) = \boldsymbol{p}^{\mathrm{T}}(\hat{\boldsymbol{x}}) \cdot \boldsymbol{a}(\boldsymbol{x}). \tag{5}$$

Define

$$J = \sum_{I=1}^{n} w(\boldsymbol{x} - \boldsymbol{x}_I)[u^h(\boldsymbol{x}, \boldsymbol{x}_I) - u_I]^2 = \sum_{I=1}^{n} w(\boldsymbol{x} - \boldsymbol{x}_I)\left[\sum_{i=1}^{m} p_i(\boldsymbol{x}_I) \cdot a_i(\boldsymbol{x}) - u_I\right]^2, \tag{6}$$

where $w(\boldsymbol{x} - \boldsymbol{x}_I)$ is a weighting function, and $\boldsymbol{x}_I$ $(I = 1, 2, \cdots, n)$ are the nodes with influence domains covering the point $\boldsymbol{x}$.

Equation (6) can be written as

$$J = (\boldsymbol{P}\boldsymbol{a} - \boldsymbol{u})^{\mathrm{T}} \boldsymbol{W}(\boldsymbol{x})(\boldsymbol{P}\boldsymbol{a} - \boldsymbol{u}), \tag{7}$$

where

$$\boldsymbol{u}^{\mathrm{T}} = (u_1, u_2, \cdots, u_n), \tag{8}$$

$$\boldsymbol{P} = \begin{bmatrix} p_1(\boldsymbol{x}_1) & p_2(\boldsymbol{x}_1) & \cdots & p_m(\boldsymbol{x}_1) \\ p_1(\boldsymbol{x}_2) & p_2(\boldsymbol{x}_2) & \cdots & p_m(\boldsymbol{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(\boldsymbol{x}_n) & p_2(\boldsymbol{x}_n) & \cdots & p_m(\boldsymbol{x}_n) \end{bmatrix}, \tag{9}$$

and

$$\boldsymbol{W}(\boldsymbol{x}) = \begin{bmatrix} w(\boldsymbol{x} - \boldsymbol{x}_1) & 0 & \cdots & 0 \\ 0 & w(\boldsymbol{x} - \boldsymbol{x}_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w(\boldsymbol{x} - \boldsymbol{x}_n) \end{bmatrix}. \tag{10}$$

From

$$\frac{\partial J}{\partial \boldsymbol{a}} = \boldsymbol{A}(\boldsymbol{x})\boldsymbol{a}(\boldsymbol{x}) - \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u} = 0, \tag{11}$$

we have

$$\boldsymbol{A}(\boldsymbol{x})\boldsymbol{a}(\boldsymbol{x}) = \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}, \tag{12}$$

where

$$\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{P}^{\mathrm{T}} \boldsymbol{W}(x) \boldsymbol{P}, \tag{13}$$

$$\boldsymbol{B}(\boldsymbol{x}) = \boldsymbol{P}^{\mathrm{T}} \boldsymbol{W}(x). \tag{14}$$

Equation (12) sometimes forms a singular or ill-conditional matrix. In order to make up for this deficiency, for basis functions

$$\boldsymbol{q} = (q_i) = (1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3, x_3x_1, \cdots), \tag{15}$$

using the Gram–Schmidt process, we can obtain

$$p_i = q_i - \sum_{k=1}^{i-1} \frac{(q_i, p_k)}{(p_k, p_k)} p_k, \ (i = 1, 2, 3, \cdots), \tag{16}$$

and

$$(p_i, p_j) = 0, \ (i \neq j). \tag{17}$$

Then, from Equation (12), $\boldsymbol{a}(\boldsymbol{x})$ can be obtained as

$$\boldsymbol{a}(\boldsymbol{x}) = \boldsymbol{A}^*(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}, \tag{18}$$

where

$$\boldsymbol{A}^*(\boldsymbol{x}) = \begin{bmatrix} \frac{1}{(p_1,p_1)} & 0 & \cdots & 0 \\ 0 & \frac{1}{(p_2,p_2)} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{(p_n,p_n)} \end{bmatrix}. \tag{19}$$

Substituting Equation (18) into Equation (5), we have

$$u^h(\boldsymbol{x}) = \sum_{I=1}^{n} \widetilde{\Phi}_I(\boldsymbol{x}) u_I = \widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}, \tag{20}$$

where

$$\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x}) = (\widetilde{\Phi}_1(\boldsymbol{x}), \widetilde{\Phi}_2(\boldsymbol{x}), \cdots, \widetilde{\Phi}_n(\boldsymbol{x})) = \boldsymbol{p}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{A}^*(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x}) \tag{21}$$

is the shape function.

This is the IMLS approximation [25], in which the shape function can be obtained more easily than the MLS approximation. Moreover, the IMLS approximation can also avoid the singular matrix. Thus, it can enhance the computational efficiency of the MLS approximation.

3. The IEFG Method for 3D Steady Convection-Diffusion-Reaction Problems

The equation of 3D steady convection-diffusion-reaction problems is considered as follows:

$$\begin{gathered} v_1(\boldsymbol{x})\frac{\partial u}{\partial x_1} + v_2(\boldsymbol{x})\frac{\partial u}{\partial x_2} + v_3(\boldsymbol{x})\frac{\partial u}{\partial x_3} - \left(k_1(\boldsymbol{x})\frac{\partial^2 u}{\partial x_1^2} + k_2(\boldsymbol{x})\frac{\partial^2 u}{\partial x_2^2} + k_3(\boldsymbol{x})\frac{\partial^2 u}{\partial x_3^2}\right) \\ +su = f(\boldsymbol{x}),\ (\boldsymbol{x} = (x_1, x_2, x_3) \in \Omega); \end{gathered} \tag{22}$$

the boundary conditions are

$$u(\boldsymbol{x}) = \overline{u},\ (\boldsymbol{x} \in \Gamma_u), \tag{23}$$

$$q(\boldsymbol{x}) = k_1(\boldsymbol{x})\frac{\partial u}{\partial x_1}n_1 + k_2(\boldsymbol{x})\frac{\partial u}{\partial x_2}n_2 + k_3(\boldsymbol{x})\frac{\partial u}{\partial x_3}n_3 = \overline{q},\ (\boldsymbol{x} \in \Gamma_q); \tag{24}$$

where $\overline{u}(\boldsymbol{x})$ and $\overline{q}(\boldsymbol{x})$ are the given values; $f(\boldsymbol{x})$ is the source term; $\Gamma = \Gamma_u \cup \Gamma_q$, $\Gamma_u \cap \Gamma_q = \varnothing$; $v_i(\boldsymbol{x})$ is the convection velocity in direction x_i; $k_i(\boldsymbol{x})$ is the diffusion efficient in the direction x_i; s is the reaction coefficient; n_i is the unit outward normal to boundary Γ in the direction x_i.

For 3D convection-diffusion-reaction equations, the equivalent functional is

$$\begin{gathered} \Pi = \int_\Omega \left[u\left(\tfrac{1}{2}su - f\right)\right]\mathrm{d}\Omega + \int_\Omega u\left(v_1\tfrac{\partial u}{\partial x_1} + v_2\tfrac{\partial u}{\partial x_2} + v_3\tfrac{\partial u}{\partial x_3}\right)\mathrm{d}\Omega \\ +\int_\Omega \tfrac{1}{2}\left[\left(\sqrt{k_1}\tfrac{\partial u}{\partial x_1}\right)^2 + \left(\sqrt{k_2}\tfrac{\partial u}{\partial x_2}\right)^2 + \left(\sqrt{k_3}\tfrac{\partial u}{\partial x_3}\right)^2\right]\mathrm{d}\Omega - \int_{\Gamma_q} u\overline{q}\mathrm{d}\Gamma. \end{gathered} \tag{25}$$

The penalty method is selected to impose the essential boundary conditions, using α to refer to the penalty factor; thus, the modified functional is

$$\Pi^* = \Pi + \frac{\alpha}{2}\int_{\Gamma_u} (u - \overline{u})(u - \overline{u})\mathrm{d}\Gamma. \tag{26}$$

Let

$$\delta\Pi^* = 0, \tag{27}$$

and the equivalent integral weak form is

$$\begin{gathered} \int_\Omega \delta u \cdot su\mathrm{d}\Omega + \int_\Omega \delta(\boldsymbol{L}u)^{\mathrm{T}} \cdot (\boldsymbol{L}u)\mathrm{d}\Omega + \int_\Omega \delta u \cdot v_1\tfrac{\partial u}{\partial x_1}\mathrm{d}\Omega + \int_\Omega \delta u \cdot v_2\tfrac{\partial u}{\partial x_2}\mathrm{d}\Omega \\ +\int_\Omega \delta u \cdot v_3\tfrac{\partial u}{\partial x_3}\mathrm{d}\Omega - \int_\Omega \delta u \cdot f\mathrm{d}\Omega - \int_{\Gamma_q} \delta u \cdot \overline{q}\mathrm{d}\Gamma + \alpha\int_{\Gamma_u} \delta u \cdot u\mathrm{d}\Gamma - \alpha\int_{\Gamma_u} \delta u \cdot \overline{u}\mathrm{d}\Gamma = 0, \end{gathered} \tag{28}$$

where

$$\boldsymbol{L}(\cdot) = \begin{bmatrix} \sqrt{k_1} \cdot \frac{\partial}{\partial x_1} \\ \sqrt{k_2} \cdot \frac{\partial}{\partial x_2} \\ \sqrt{k_3} \cdot \frac{\partial}{\partial x_3} \end{bmatrix} (\cdot). \tag{29}$$

We select M nodes $\boldsymbol{x}_I$ in the 3D domain Ω; therefore, the corresponding function value is

$$u(\boldsymbol{x}_I) = u_I. \tag{30}$$

From Section 2, we have

$$u(\boldsymbol{x}) = \widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u} = \sum_{I=1}^{n} \widetilde{\Phi}_I(\boldsymbol{x}) u_I, \tag{31}$$

the form of vector $\boldsymbol{u}$ is

$$\boldsymbol{u} = (u_1, u_2, \cdots, u_n)^{\mathrm{T}}. \tag{32}$$

Thus, we have

$$\boldsymbol{L}u(\boldsymbol{x}) = \sum_{I=1}^{n} \begin{bmatrix} \sqrt{k_1} \cdot \frac{\partial}{\partial x_1} \\ \sqrt{k_2} \cdot \frac{\partial}{\partial x_2} \\ \sqrt{k_3} \cdot \frac{\partial}{\partial x_3} \end{bmatrix} \widetilde{\Phi}_I(\boldsymbol{x}) u_I = \sum_{I=1}^{n} \boldsymbol{B}_I(\boldsymbol{x}) u_I = \boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}, \tag{33}$$

where

$$\boldsymbol{B}(\boldsymbol{x}) = (\boldsymbol{B}_1(\boldsymbol{x}), \boldsymbol{B}_2(\boldsymbol{x}), \cdots, \boldsymbol{B}_n(\boldsymbol{x})), \tag{34}$$

$$\boldsymbol{B}_I(x) = \begin{bmatrix} \sqrt{k_1} \cdot \widetilde{\Phi}_{I,1}(\boldsymbol{x}) \\ \sqrt{k_2} \cdot \widetilde{\Phi}_{I,2}(\boldsymbol{x}) \\ \sqrt{k_3} \cdot \widetilde{\Phi}_{I,3}(\boldsymbol{x}) \end{bmatrix}. \tag{35}$$

Substituting Equations (31) and (33) into Equation (28), we have

$$\begin{gathered} s\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot [\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega + \int_\Omega \delta[\boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot [\boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega + \\ v_1\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \frac{\partial}{\partial x_1}[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega + v_2\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \frac{\partial}{\partial x_2}[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega + \\ v_3\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \frac{\partial}{\partial x_3}[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega - \int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot f\mathrm{d}\Omega - \int_{\Gamma_q} \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \overline{q}\mathrm{d}\Gamma \\ +\alpha\int_{\Gamma_u} \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot [\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Gamma - \alpha\int_{\Gamma_u} \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \overline{u}\mathrm{d}\Gamma = 0. \end{gathered} \tag{36}$$

All integral terms of Equation (36) are analyzed as follows:

$$s\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot [\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega = \delta\boldsymbol{u}^{\mathrm{T}} \cdot [s\int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Omega] \cdot \boldsymbol{u} = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{C}_s \cdot \boldsymbol{u}, \tag{37}$$

$$\int_\Omega \delta[\boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot [\boldsymbol{B}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega = \delta\boldsymbol{u}^{\mathrm{T}} \cdot [\int_\Omega \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x})\mathrm{d}\Omega] \cdot \boldsymbol{u} = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{K} \cdot \boldsymbol{u}, \tag{38}$$

$$\begin{gathered} v_1\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \frac{\partial}{\partial x_1}[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega = \delta\boldsymbol{u}^{\mathrm{T}} \cdot v_1[\int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x}) \cdot \frac{\partial}{\partial x_1}\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Omega] \cdot \boldsymbol{u} \\ = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{G}_1 \cdot \boldsymbol{u}, \end{gathered} \tag{39}$$

$$v_2\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \frac{\partial}{\partial x_2}[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega = \delta\boldsymbol{u}^{\mathrm{T}} \cdot v_2[\int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x}) \cdot \frac{\partial}{\partial x_2}\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Omega] \cdot \boldsymbol{u} = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{G}_2 \cdot \boldsymbol{u} \tag{40}$$

$$\begin{gathered} v_3\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \frac{\partial}{\partial x_3}[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Omega = \delta\boldsymbol{u}^{\mathrm{T}} \cdot v_3[\int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x}) \cdot \frac{\partial}{\partial x_3}\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Omega] \cdot \boldsymbol{u} \\ = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{G}_3 \cdot \boldsymbol{u}, \end{gathered} \tag{41}$$

$$\int_\Omega \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot f\mathrm{d}\Omega = \delta\boldsymbol{u}^{\mathrm{T}} \cdot [\int_{\Omega^{(k)}} \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})f\mathrm{d}\Omega] = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{F}_1, \tag{42}$$

$$\int_{\Gamma_q} \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \overline{q}\mathrm{d}\Gamma = \delta\boldsymbol{u}^{\mathrm{T}} \cdot [\int_{\Gamma_q} \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\overline{q}\mathrm{d}\Gamma] = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{F}_q, \tag{43}$$

$$\alpha \int_{\Gamma_u} \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot [\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]\mathrm{d}\Gamma = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \alpha[\int_{\Gamma_u} \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\widetilde{\boldsymbol{\Phi}}(x)\mathrm{d}\Gamma] \cdot \boldsymbol{u} = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{K}_\alpha \cdot \boldsymbol{u}, \tag{44}$$

$$\alpha \int_{\Gamma_u} \delta[\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\boldsymbol{u}]^{\mathrm{T}} \cdot \overline{\boldsymbol{u}}\mathrm{d}\Gamma = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \alpha[\int_{\Gamma_u} \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\overline{\boldsymbol{u}}\mathrm{d}\Gamma] = \delta\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{F}_\alpha, \tag{45}$$

where

$$\boldsymbol{C}_s = s\int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Omega, \tag{46}$$

$$\boldsymbol{K} = \int_\Omega \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{B}(\boldsymbol{x})\mathrm{d}\Omega, \tag{47}$$

$$\boldsymbol{K}_\alpha = \alpha\int_{\Gamma_u} \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Gamma, \tag{48}$$

$$\boldsymbol{G}_1 = v_1\int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\frac{\partial}{\partial x_1}\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Omega, \tag{49}$$

$$\boldsymbol{G}_2 = v_2\int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\frac{\partial}{\partial x_2}\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Omega, \tag{50}$$

$$\boldsymbol{G}_3 = v_3\int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\frac{\partial}{\partial x_3}\widetilde{\boldsymbol{\Phi}}(\boldsymbol{x})\mathrm{d}\Omega, \tag{51}$$

$$\boldsymbol{F}_1 = \int_\Omega \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})f\mathrm{d}\Omega, \tag{52}$$

$$\boldsymbol{F}_q = \int_{\Gamma_q} \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\overline{q}\mathrm{d}\Gamma, \tag{53}$$

$$\boldsymbol{F}_\alpha = \alpha\int_{\Gamma_u} \widetilde{\boldsymbol{\Phi}}^{\mathrm{T}}(\boldsymbol{x})\overline{u}\mathrm{d}\Gamma. \tag{54}$$

Substituting Equations (37)–(45) into Equation (36), we have

$$\delta\boldsymbol{u}^{\mathrm{T}} \cdot (\boldsymbol{C}_s\boldsymbol{u} + \boldsymbol{K}\boldsymbol{u} + \boldsymbol{K}_\alpha\boldsymbol{u} + \boldsymbol{G}_1\boldsymbol{u} + \boldsymbol{G}_2\boldsymbol{u} + \boldsymbol{G}_3\boldsymbol{u} - \boldsymbol{F}_1 - \boldsymbol{F}_q - \boldsymbol{F}_\alpha) = 0. \tag{55}$$

Let

$$\hat{\boldsymbol{K}} = \boldsymbol{C}_s + \boldsymbol{K} + \boldsymbol{K}_\alpha + \boldsymbol{G}_1 + \boldsymbol{G}_2 + \boldsymbol{G}_3, \tag{56}$$

$$\hat{\boldsymbol{F}} = \boldsymbol{F}_1 + \boldsymbol{F}_q + \boldsymbol{F}_\alpha. \tag{57}$$

Because $\delta\boldsymbol{u}^{\mathrm{T}}$ is arbitrary, Equation (55) can be transformed as

$$\hat{\boldsymbol{K}}\boldsymbol{u} = \hat{\boldsymbol{F}}. \tag{58}$$

This is the IEFG method for 3D convection-diffusion-reaction problems.

4. Numerical Examples

We compute the relative error of the IEFG method for 3D convection-diffusion-reaction problems; thus, the formula is given as

$$\|u - u^h\|^{rel}_{L^2(\Omega)} = \frac{\left(\int_\Omega (u - u^h)^2\mathrm{d}\Omega\right)^{1/2}}{\|u\|_{L^2(\Omega)}}. \tag{59}$$

The IEFG and the EFG methods are used to solve these numerical examples. We select regularly distributed nodes, $3 \times 3 \times 3$ Gaussian points are selected in each integral cell, and the linear basis function is selected. The relative error and calculation efficiency of the IEFG and the EFG methods are compared.

The first example is

$$\begin{array}{l}(1+\cos x_1)\frac{\partial u}{\partial x_1}+(1+\cos x_2)\frac{\partial u}{\partial x_2}+(1+\cos x_3)\frac{\partial u}{\partial x_3}+\\ (1+x_1^4)\frac{\partial^2 u}{\partial x_1^2}+(1+x_2^4)\frac{\partial^2 u}{\partial x_2^2}+(1+x_3^4)\frac{\partial^2 u}{\partial x_3^2}-u=f(x),\end{array} \tag{60}$$

and the problem domain is Ω = [0,0.5] × [0,0.5] × [0,0.5].

The boundary conditions are

$$u\big|_{x_1=0}=1+\cosh(x_2)+\cosh(x_3), \tag{61}$$

$$u\big|_{x_1=0.5}=\cosh(0.5)+\cosh(x_2)+\cosh(x_3), \tag{62}$$

$$u\big|_{x_2=0}=\cosh(x_1)+1+\cosh(x_3), \tag{63}$$

$$u\big|_{x_2=0.5}=\cosh(x_1)+\cosh(0.5)+\cosh(x_3), \tag{64}$$

$$u\big|_{x_3=0}=\cosh(x_1)+\cosh(x_2)+1, \tag{65}$$

$$u\big|_{x_3=0.5}=\cosh(x_1)+\cosh(x_3)+\cosh(0.5). \tag{66}$$

The theoretical result is

$$u=\cosh(x_1)+\cosh(x_2)+\cosh(x_3). \tag{67}$$

In this paper, we should discuss the scale parameter, penalty factor, and convergence.

First, different values of $d_{\max}$ will influence the relative error of the numerical solution. In addition, 11 × 11 × 11 regular nodes, 10 × 10 × 10 integral cells, and the cubic spline function are used, and $\alpha = 7.0 \times 10^3$. Figure 1 shows the relationship between $d_{\max}$ and the relative error. We can see that if $d_{\max}$ = 1.4~1.5, the relative error is smaller.

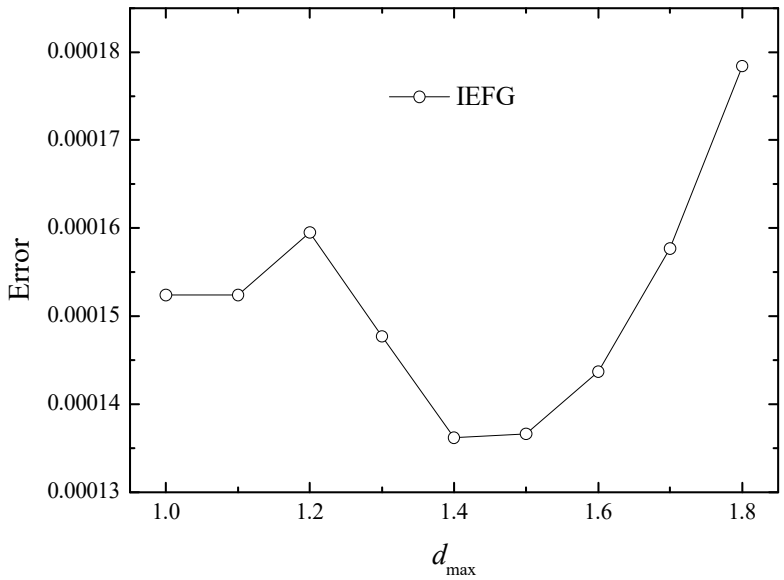

Figure 1. The relationship between $d_{\max}$ and the relative error.

Secondly, different values of α will influence the relative error of the numerical solution. In addition, 11 × 11 × 11 regular nodes, 10 × 10 × 10 integral cells, and the cubic spline function are used, and $d_{\max}$ = 1.44. Figure 2 shows the relationship between α and the relative error. We can see that when $\alpha = 5.0 \times 10^3$~1.0×10^4, the relative error is smaller.

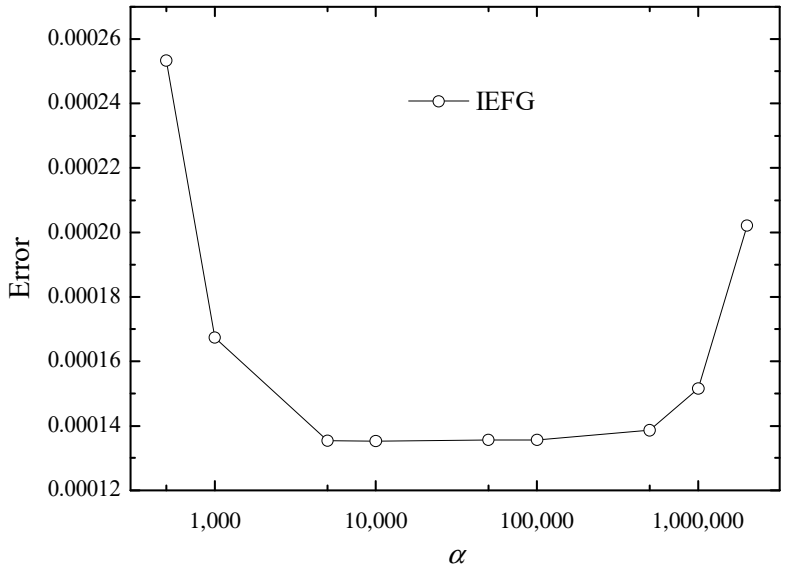

Figure 2. The relationship between α and the relative error.

In order to demonstrate the convergence of the IEFG method, the distribution of nodes must be discussed. $d_{\max} = 1.44$, $\alpha = 7.0 \times 10^3$, and the cubic spline function is used. Table 1 shows the relationship between the node distribution and the relative errors. It is easy to see that with increase in nodes, the relative error shows a decreasing trend. Figure 3 shows the contour plot of nodes, numerical solutions, and the relative errors.

Table 1. The relative errors and CPU times of the IEFG and the EFG methods with the increase in node distribution.

Nodes	Relative Error		Time (s)	
	IEFG	EFG	IEFG	EFG
$3 \times 3 \times 3$	0.36176%	0.36176%	0.47	0.50
$5 \times 5 \times 5$	0.09398%	0.09398%	3.61	3.80
$7 \times 7 \times 7$	0.03752%	0.03752%	12.2	13.0
$9 \times 9 \times 9$	0.01865%	0.01865%	30.6	32.5
$11 \times 11 \times 11$	0.01352%	0.01352%	61.7	64.7
$13 \times 13 \times 13$	0.01388%	0.01388%	122.1	126.2
$15 \times 15 \times 15$	0.01539%	0.01539%	229.0	236.6
$17 \times 17 \times 17$	0.01682%	0.01682%	371.6	396.9
$19 \times 19 \times 19$	0.01798%	0.01798%	609.3	643.7
$21 \times 21 \times 21$	0.01889%	0.01889%	955.7	1007.7

In this example, we select $11 \times 11 \times 11$ regular nodes, $10 \times 10 \times 10$ integral cells, and the cubic spline function. In order to obtain the numerical results with smaller relative errors by using the IEFG and the EFG methods, we should select the appropriate parameters in MATLAB codes, $d_{\max} = 1.44$ and $\alpha = 7.0 \times 10^3$; thus, the smaller relative errors of the IEFG and the EFG methods are equal to 0.0135%, and the corresponding CPU times are 61.7 s and 64.7 s, respectively.

Figures 4–6 and Tables 2–4 show the comparison of the numerical solutions of two methods and the analytical ones. We can see that the numerical solutions of both methods are in good agreement with the analytical one.

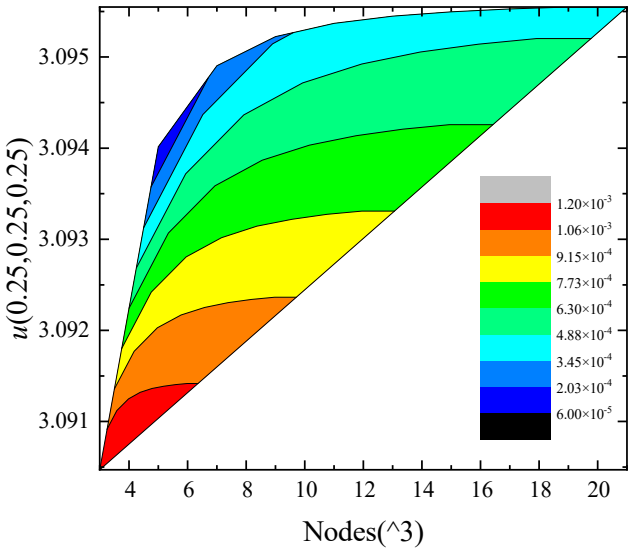

Figure 3. The contour plot of nodes, numerical solutions, and the relative errors.

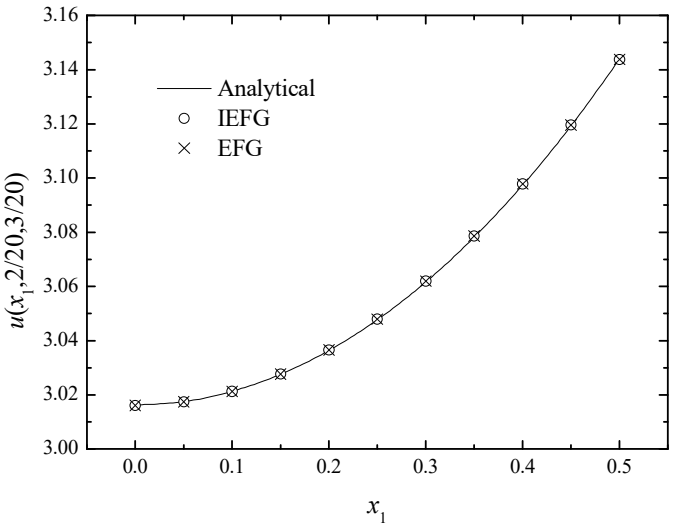

Figure 4. The comparison of numerical and exact solutions along x_1-axis.

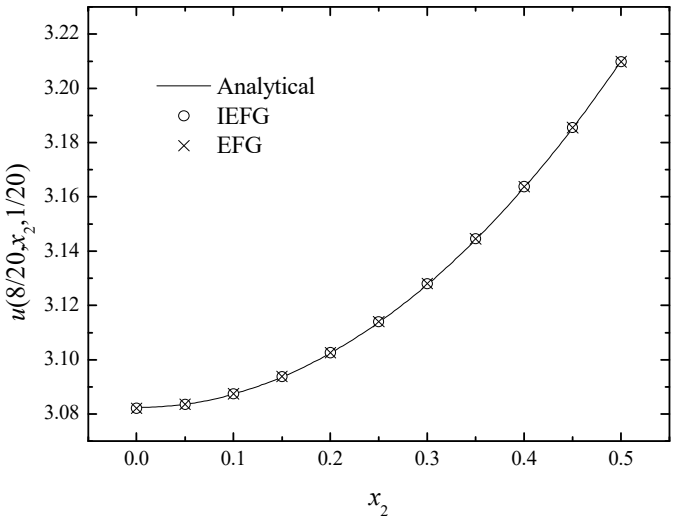

Figure 5. The comparison of numerical and exact solutions along x_2-axis.

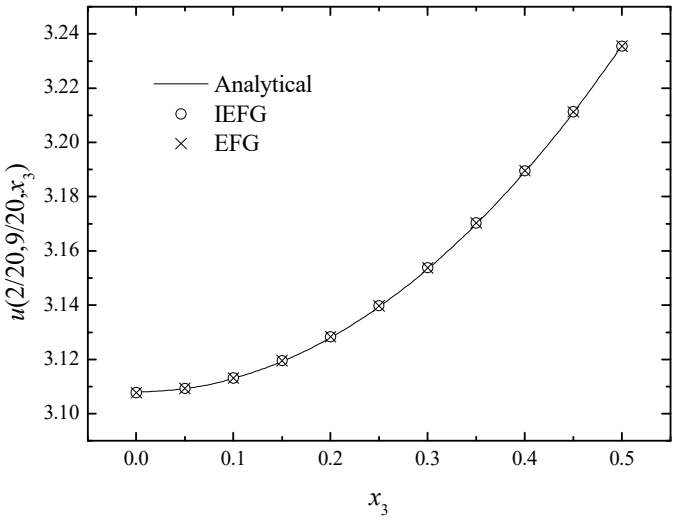

Figure 6. The comparison of numerical and exact solutions along x_3-axis.

Table 2. The analytical and numerical solutions of $u(x_1, 2/20, 3/20)$.

x_1	Analytical	IEFG	EFG
0	3.01628	3.01607	3.01607
0.05	3.01753	3.01740	3.01740
0.1	3.02128	3.02129	3.02129
0.15	3.02755	3.02767	3.02767
0.2	3.03634	3.03657	3.03657
0.25	3.04769	3.04801	3.04801
0.3	3.06161	3.06202	3.06202
0.35	3.07815	3.07863	3.07863
0.4	3.09735	3.09782	3.09782
0.45	3.11925	3.11955	3.11955
0.5	3.14390	3.14377	3.14377

Table 3. The exact and numerical solutions of $u(8/20, x_2, 1/20)$.

x_2	Analytical	IEFG	EFG
0	3.08232	3.08211	3.08211
0.05	3.08357	3.08355	3.08355
0.1	3.08733	3.08745	3.08745
0.15	3.09359	3.09379	3.09379
0.2	3.10239	3.10264	3.10264
0.25	3.11374	3.11403	3.11403
0.3	3.12766	3.12801	3.12801
0.35	3.1442	3.14459	3.14459
0.4	3.1634	3.16378	3.16378
0.45	3.18529	3.18555	3.18555
0.5	3.20995	3.20982	3.20982

Table 4. The exact and numerical solutions of $u(2/20, 9/20, x_3)$.

x_3	Analytical	IEFG	EFG
0	3.10797	3.10774	3.10774
0.05	3.10922	3.1093	3.1093
0.1	3.11298	3.11321	3.11321
0.15	3.11925	3.11955	3.11955
0.2	3.12804	3.12839	3.12839
0.25	3.13939	3.13979	3.13979
0.3	3.15331	3.15376	3.15376
0.35	3.16985	3.17033	3.17033
0.4	3.18905	3.18952	3.18952
0.45	3.21094	3.21128	3.21128
0.5	3.2356	3.23545	3.23545

When using the IEFG method to solve it, calculation resources can be saved although similar relative errors are obtained.

In addition, singular matrices can be avoided if the IEFG method is selected. When the EFG method is used, we select $d_{\max} = 1.0$; thus, the singular matrices appear. If the IEFG method is selected, $\alpha = 7.0 \times 10^3$, $d_{\max} = 1.0$, and the relative error is 0.0152%. The numerical solutions and analytical ones are compared in Figure 7. We can see that numerical solutions are in good agreement with the analytical one.

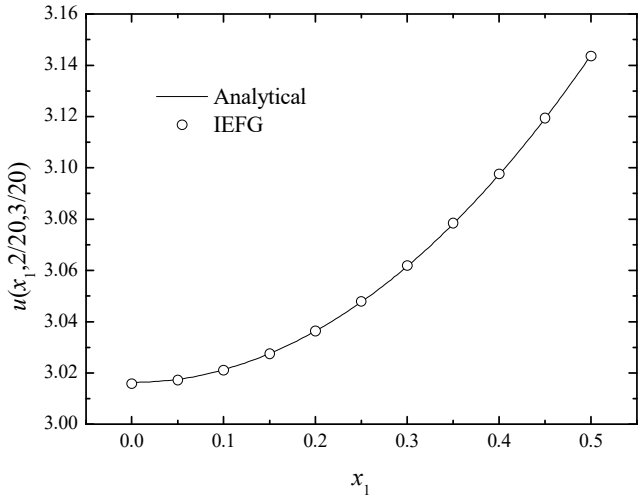

Figure 7. The comparison of numerical and exact solutions along x_1-axis.

The second example [11] is

$$10x_1\frac{\partial u}{\partial x_1} + 10x_2\frac{\partial u}{\partial x_2} + 10x_3\frac{\partial u}{\partial x_3} + \Delta u - [10(x_1 + x_2 + x_3) + 3]u = 0, \tag{68}$$

and the problem domain is $\Omega = [0,1] \times [0,1] \times [0,1]$.

The boundary conditions are

$$u\Big|_{x_1=0} = e^{0+x_2+x_3}, \tag{69}$$

$$u\Big|_{x_1=1} = e^{1+x_2+x_3}, \tag{70}$$

$$u\Big|_{x_2=0} = e^{0+x_1+x_3}, \tag{71}$$

$$u\Big|_{x_2=1} = e^{1+x_1+x_3}, \tag{72}$$

$$u\Big|_{x_3=0} = e^{0+x_2+x_1}, \tag{73}$$

$$u\Big|_{x_3=1} = e^{1+x_2+x_1}. \tag{74}$$

The theoretical result is

$$u = e^{x_1+x_2+x_3}. \tag{75}$$

In this example, we select $11 \times 11 \times 11$ regular nodes, $10 \times 10 \times 10$ integral cells, and the cubic spline function. In order to obtain the numerical results with smaller relative errors by using the IEFG and the EFG methods, we should select the appropriate parameters in MATLAB codes, $\alpha = 7.0 \times 10^2$, $d_{\max} = 1.21$; thus, the smaller relative errors of the IEFG and the EFG methods are equal to 0.0892%, and the corresponding CPU times are 52.8 s and 55.8 s.

Figures 8–10 show the comparison of the numerical solutions of two methods and the analytical ones. We can see that the numerical solutions of both methods are in good agreement with the analytical one.

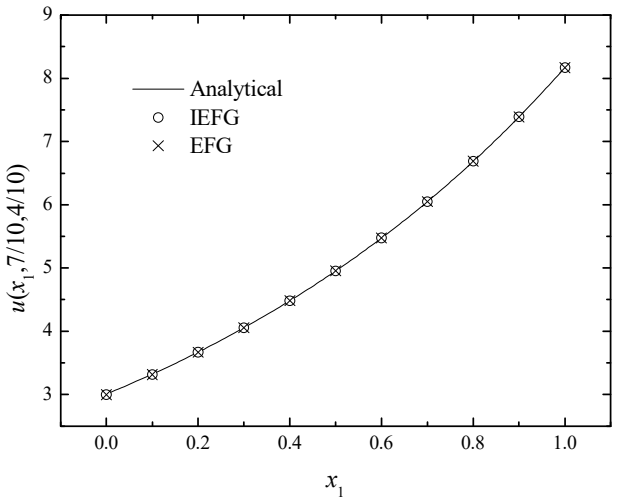

Figure 8. The comparison of numerical and exact solutions along x_1-axis.

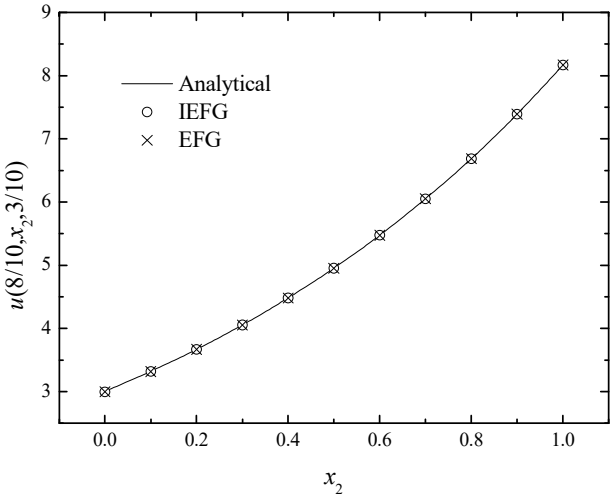

Figure 9. The comparison of numerical and exact solutions along x_2-axis.

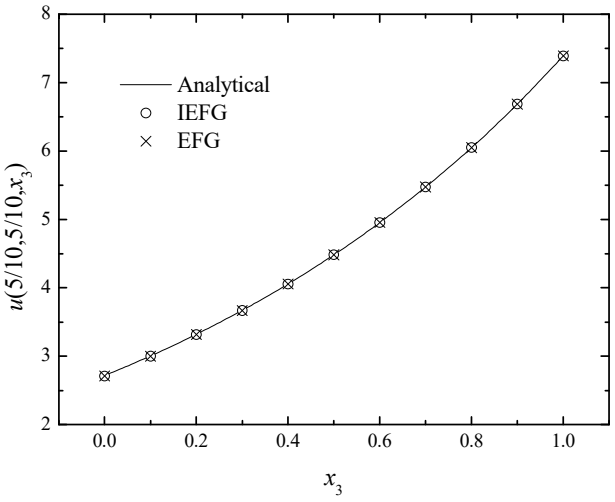

Figure 10. The comparison of numerical and exact solutions along x_3-axis.

From this example, we can see that, under the condition of similar calculation precision, the IEFG method can solve 3D convection-diffusion-reaction problems successfully with less calculation resources.

In addition, singular matrices can be avoided if the IEFG method is selected. When the EFG method is used, we select $d_{\max} = 1.0$; thus, the singular matrices appear. If the IEFG method is selected, $\alpha = 7.0 \times 10^2$, $d_{\max} = 1.0$, and the relative error is 0.0922%. The numerical solutions and analytical ones are compared in Figure 11. We can see that numerical solutions are in good agreement with the analytical one.

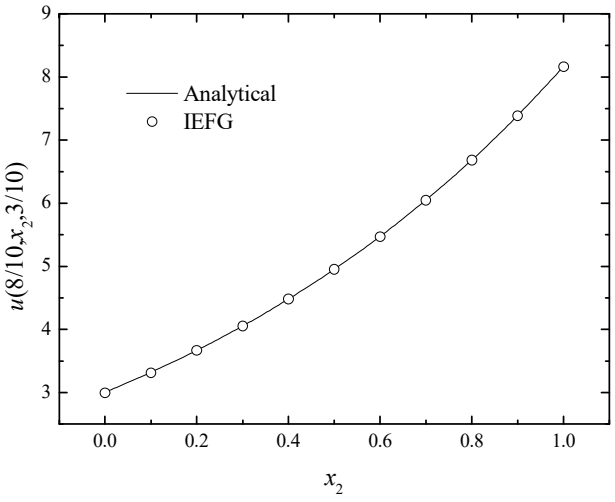

Figure 11. The comparison of numerical and exact solutions along x_2-axis.

The third example [44–46] is

$$\lambda(\boldsymbol{x})\frac{\partial u}{\partial x_1} + \mu(\boldsymbol{x})\frac{\partial u}{\partial x_2} + \nu(\boldsymbol{x})\frac{\partial u}{\partial x_3} + \Delta u = f(\boldsymbol{x}), \tag{76}$$

where

$$\lambda(\boldsymbol{x}) = \mathrm{Re}x_1(1-2x_2)(1-x_3), \tag{77}$$

$$\mu(\boldsymbol{x}) = \mathrm{Re}x_2(1-2x_3)(1-x_1), \tag{78}$$

$$\nu(\boldsymbol{x}) = \mathrm{Re}x_3(1-2x_1)(1-x_2), \tag{79}$$

and the problem domain is $\Omega = [0,0.5] \times [0,0.5] \times [0,0.5]$.

The boundary conditions are

$$u\big|_{x_1=0} = u\big|_{x_1=0.5} = \sin(\pi x_2) + \sin(\pi x_3) + \sin(3\pi x_2) + \sin(3\pi x_3), \tag{80}$$

$$u\big|_{x_2=0} = u\big|_{x_2=0.5} = \sin(\pi x_1) + \sin(\pi x_3) + \sin(3\pi x_1) + \sin(3\pi x_3), \tag{81}$$

$$u\big|_{x_3=0} = u\big|_{x_3=0.5} = \sin(\pi x_1) + \sin(\pi x_2) + \sin(3\pi x_1) + \sin(3\pi x_2). \tag{82}$$

The theoretical result is

$$u = \sin(\pi x_1) + \sin(\pi x_2) + \sin(\pi x_3) + \sin(3\pi x_1) + \sin(3\pi x_2) + \sin(3\pi x_3). \tag{83}$$

In this example, we select Re = 100, 19 × 19 × 19 regular nodes, 18 × 18 × 18 integral cells, and the cubic spline function. In order to obtain the numerical results with smaller relative errors by using the IEFG and the EFG methods, we should select the appropriate parameters in MATLAB codes, $d_{\max} = 1.29$, $\alpha = 1.6 \times 10^3$; thus, the smaller relative errors of the IEFG and the EFG methods are equal to 0.2646%, and the corresponding CPU times are 497.7 s and 525.8 s.

Figures 12–14 show the comparison of the numerical solutions of two methods and the analytical ones. We can see that the numerical solutions of both methods are in good agreement with the analytical one.

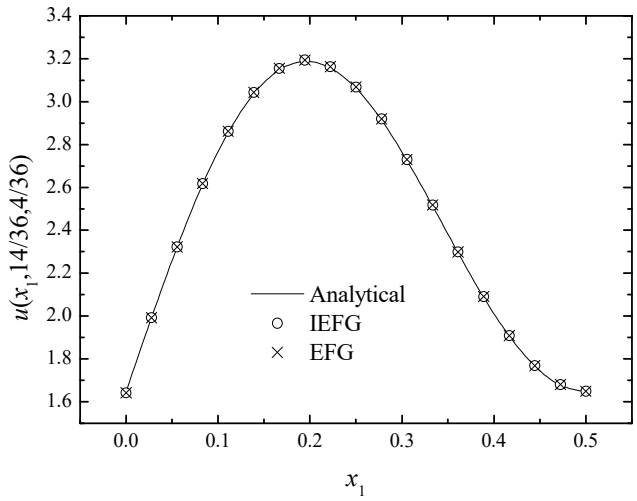

Figure 12. The comparison of numerical and exact solutions along x_1-axis.

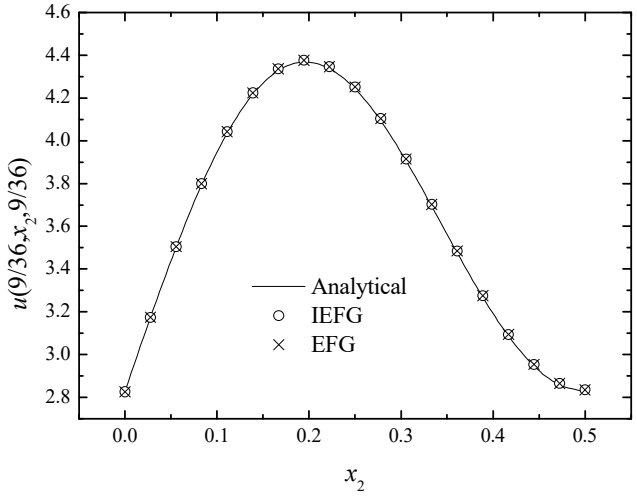

Figure 13. The comparison of numerical and exact solutions along x_2-axis.

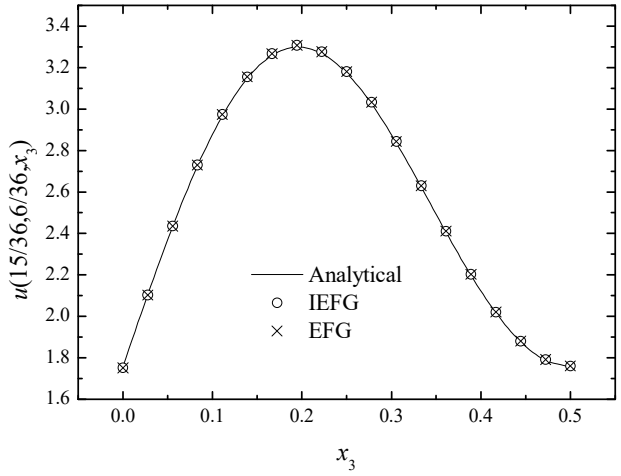

Figure 14. The comparison of numerical and exact solutions along x_3-axis.

From this example, we can see that, under the condition of similar calculation precision, the IEFG method can solve 3D convection-diffusion-reaction problems successfully with less calculation resources.

In addition, singular matrices can be avoided if the IEFG method is selected. When the EFG method is used, we select $d_{\max} = 1.0$; thus, the singular matrices appear. If the IEFG method is selected, $\alpha = 1.6 \times 10^3$, $d_{\max} = 1.0$, and the relative error is 0.2659%. The numerical solutions and analytical ones are compared in Figure 15. We can see that numerical solutions are in good agreement with the analytical one.

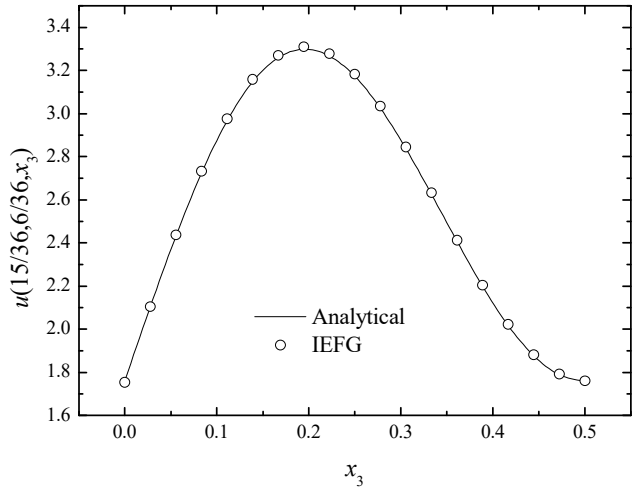

Figure 15. The comparison of numerical and exact solutions along x_3-axis.

The fourth example [46] is

$$\frac{1}{x_1}\frac{\partial u}{\partial x_1} + \frac{\partial^2 u}{\partial x_1^2} + \frac{1}{x_1^2}\frac{\partial^2 u}{\partial x_2^2} + \frac{\partial^2 u}{\partial x_3^2} = f(x_1, x_2, x_3), \tag{84}$$

and the problem domain is $\Omega = [0,1] \times [0,1] \times [0,1]$.

The boundary conditions are

$$u\big|_{x_1=0} = 0, \tag{85}$$

$$u\big|_{x_1=1} = \cos(\pi x_2)\cos(\pi x_3), \tag{86}$$

$$u\Big|_{x_2=0} = x_1^2 \cos(\pi x_3), \tag{87}$$

$$u\Big|_{x_2=1} = -x_1^2\cos(\pi x_3), \tag{88}$$

$$u\Big|_{x_3=0} = x_1^2\cos(\pi x_2), \tag{89}$$

$$u\Big|_{x_3=1} = -x_1^2\cos(\pi x_2). \tag{90}$$

The theoretical result is

$$u = x_1^2\cos(\pi x_2)\cos(\pi x_3). \tag{91}$$

In this example, we select $15 \times 15 \times 15$ regular nodes, $14 \times 14 \times 14$ integral cells, and the cubic spline function. In order to obtain the numerical results with smaller relative errors by using the IEFG and the EFG methods, we should select the appropriate parameters in MATLAB codes, $d_{\max} = 1.21$, $\alpha = 2.9 \times 10^4$; thus, the smaller relative errors of the IEFG and the EFG methods are equal to 0.5147%, and the corresponding CPU times are 111.2 s and 118.3 s.

Figures 16–18 show the comparison of the numerical solutions of the two methods and the analytical ones. We can see that the numerical solutions of both methods are in good agreement with the analytical one.

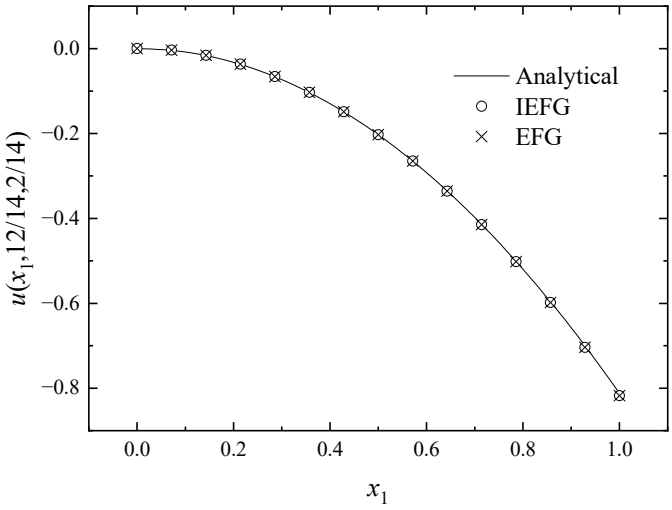

Figure 16. The comparison of numerical and exact solutions along x_1-axis.

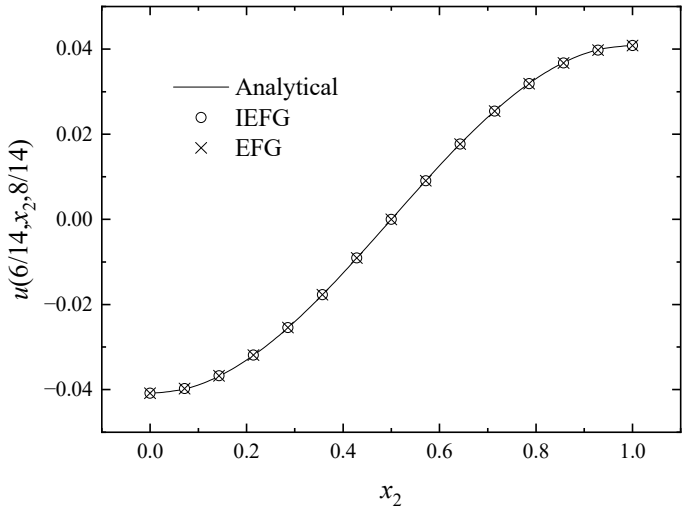

Figure 17. The comparison of numerical and exact solutions along x_2-axis.

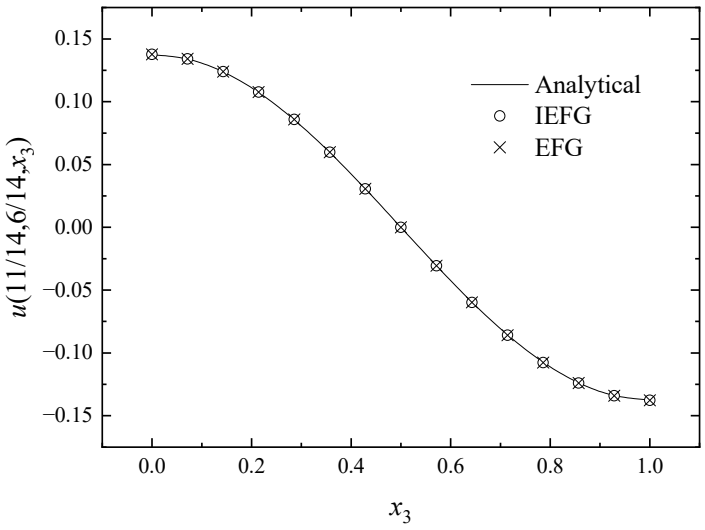

Figure 18. The comparison of numerical and exact solutions along x_3-axis.

We can see that the calculation efficiency can be improved when using the IEFG method to analyze it.

In addition, singular matrices can be avoided if the IEFG method is selected. When the EFG method is used, we select $d_{\max} = 1.0$; thus, the singular matrices appear. If the IEFG method is selected, $\alpha = 2.9 \times 10^4$, $d_{\max} = 1.0$, and the relative error is 0.5346%. The numerical solutions and analytical ones are compared in Figure 19. We can see that numerical solutions are in good agreement with the analytical one.

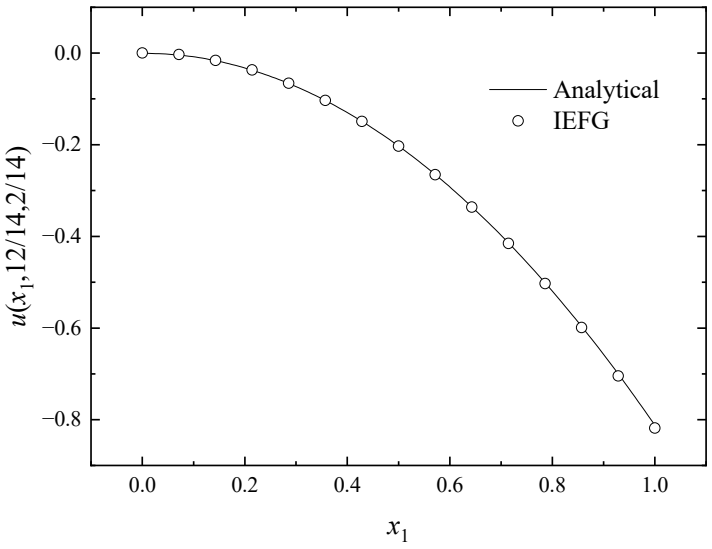

Figure 19. The comparison of numerical and exact solutions along x_1-axis.

5. Conclusions

In this paper, we select the IEFG method instead of the traditional EFG method to solve 3D convection-diffusion-reaction problems with variable coefficients.

The convergence is demonstrated numerically from numerical examples, the correctness of the IEFG method is verified, and we can see that the IEFG method can improve the calculation speed of the EFG method without losing calculation accuracy. Additionally, the IEFG method can avoid singular matrices that often exist in the EFG method.

Our study can extend the scope of application of the IEFG method in science and engineering fields.

Author Contributions: Conceptualization, H.C.; methodology, H.C.; software, H.C.; writing—original draft preparation, H.C.; writing—review and editing, Z.X. and Y.L.; visualization, Z.X.; supervision, H.C.; funding acquisition, H.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of Shanxi Province (Grant No. 20210302124388).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hidayat, M. Meshless finite difference method with B-splines for numerical solution of coupled advection-diffusion-reaction problems. *Int. J. Therm. Sci.* **2021**, *165*, 106933. [CrossRef]
2. Abdulle, A.; Souza, G. A local adaptive discontinuous Galerkin method for convection-diffusion-reaction equations. *J. Comput. Phys.* **2022**, *451*, 110894. [CrossRef]
3. Mesgarani, H.; Kermani, M.; Abbaszadeh, M. Application of SPD-RBF method of lines for solving nonlinear advection-diffusion-reaction equation with variable coefficients. *Int. J. Numer. Methods Heat Fluid Flow* **2022**, *32*, 850–886. [CrossRef]
4. Al-Bayati, S.A.; Wrobel, L.C. Numerical modelling of convection-diffusion problems with first-order chemical reaction using the dual reciprocity boundary element method. *Int. J. Numer. Methods Heat Fluid Flow* **2022**, *32*, 1793–1823. [CrossRef]
5. Chen, L.; Cheng, Y.M. Reproducing kernel particle method with complex variables for elasticity. *Acta Phys. Sin.* **2008**, *57*, 1–10. [CrossRef]
6. Dai, B.D.; Cheng, Y.M. Local boundary integral equation method based on radial basis functions for potential problems. *Acta Phys. Sin.* **2007**, *56*, 597–603.
7. Cheng, H.; Peng, M.J.; Cheng, Y.M. A hybrid improved complex variable element-free Galerkin method for three-dimensional advection-diffusion problems. *Eng. Anal. Bound. Elem.* **2018**, *97*, 39–54. [CrossRef]
8. Sun, F.X.; Wang, J.F.; Cheng, Y.M. An improved interpolating element-free Galerkin method for elastoplasticity via nonsingular weight functions. *Int. J. Appl. Mech.* **2016**, *8*, 1650096. [CrossRef]
9. Wu, Q.; Peng, M.J.; Cheng, Y.M. The interpolating dimension splitting element-free Galerkin method for 3D potential problems. *Eng. Comput.* **2022**, *38*, 2703–2717. [CrossRef]
10. Peng, P.P.; Cheng, Y.M. Analyzing three-dimensional wave propagation with the hybrid reproducing kernel particle method based on the dimension splitting method. *Eng. Comput.* **2022**, *38*, 1131–1147. [CrossRef]
11. Romão, E.C.; de Moura, L.F.M. Galerkin and Least Squares Methods to Solve a 3D Convection-Diffusion-Reaction Equation with Variable Coefficients. *Numer. Heat Transf. Part A Appl.* **2012**, *61*, 669–698. [CrossRef]
12. Zhang, X.H.; Xiang, H. Variational multiscale element free Galerkin method for convection-diffusion-reaction equation with small diffusion. *Eng. Anal. Bound. Elem.* **2014**, *46*, 85–92. [CrossRef]
13. Zhang, T.; Li, X.L. A variational multiscale interpolating element-free Galerkin method for convection-diffusion and Stokes problems. *Eng. Anal. Bound. Elem.* **2017**, *82*, 185–193. [CrossRef]
14. Cao, X.T.; Zhang, X.H.; Shi, X.T. An adaptive variational multiscale element free Galerkin method based on the residual-based a posteriori error estimators for convection-diffusion-reaction problems. *Eng. Anal. Bound. Elem.* **2022**, *136*, 238–251. [CrossRef]
15. Li, J.W.; Qiao, Y.Y.; Zhai, S.Y.; Feng, X.L. Meshless local Petrov Galerkin method for 2D/3D nonlinear convection-diffusion equations based on LS-RBF-PUM. *Numer. Heat Transf. Part B Fundam.* **2018**, *74*, 450–464. [CrossRef]
16. Li, J.W.; Feng, X.L.; He, Y.N. RBF-based meshless local Petrov Galerkin method for the multi-dimensional convection-diffusion-reaction equation. *Eng. Anal. Bound. Elem.* **2019**, *98*, 46–53. [CrossRef]
17. Chang, J.Y.; Chen, R.Y.; Tsai, C.C. Hermite method of approximate particular solutions for solving time-dependent convection-diffusion-reaction problems. *Mathematics* **2022**, *10*, 188. [CrossRef]
18. Wang, F.J.; Wang, C.; Chen, Z.T. Local knot method for 2D and 3D convection-diffusion-reaction equations in arbitrary domains. *Appl. Math. Lett.* **2020**, *105*, 106308. [CrossRef]
19. Belytschko, T.; Lu, Y.Y.; Gu, L. Element-free Galerkin Methods. *Int. J. Numer. Methods Eng.* **1994**, *37*, 229–256. [CrossRef]
20. Lancaster, P.; Salkauskas, K. Surfaces generated by moving least squares methods. *Math. Comput.* **1981**, *37*, 141–158. [CrossRef]
21. Cheng, J. Mathematical models and data analysis of residential land leasing behavior of district governments of Beijing in China. *Mathematics* **2021**, *9*, 2314. [CrossRef]
22. Cheng, J.; Xie, Y.; Zhang, J. Industry structure optimization via the complex network of industry space: A case study of Jiangxi Province in China. *J. Clean. Prod.* **2022**, *338*, 130602. [CrossRef]
23. Cheng, J.; Luo, X.W. Analyzing the land leasing behavior of the government of Beijing, China, via the multinomial logit model. *Land* **2022**, *11*, 376. [CrossRef]
24. Cheng, J. Residential land leasing and price under public land ownership. *J. Urban Plan. Dev.* **2021**, *147*, 0502100. [CrossRef]
25. Cheng, Y.M.; Chen, M.J. A boundary element-free method for linear elasticity. *Acta Mech. Sin.* **2003**, *35*, 181–186.

26. Zhang, Z.; Zhao, P.; Liew, K.M. Analyzing three-dimensional potential problems with the improved element-free Galerkin method. *Comput. Mech.* **2009**, *44*, 273–284. [CrossRef]
27. Yu, S.Y.; Peng, M.J.; Cheng, H.; Cheng, Y.M. The improved element-free Galerkin method for three-dimensional elastoplasticity problems. *Eng. Anal. Bound. Elem.* **2019**, *104*, 215–224. [CrossRef]
28. Zheng, G.D.; Cheng, Y.M. The improved element-free Galerkin method for diffusional drug release problems. *Int. J. Appl. Mech.* **2020**, *12*, 2050096. [CrossRef]
29. Cheng, H.; Zheng, G.D. Analyzing 3D advection-diffusion problems by using the improved element-free Galerkin method. *Math. Probl. Eng.* **2020**, *2020*, 4317538. [CrossRef]
30. Cheng, H.; Peng, M.J. The improved element-free Galerkin method for 3D Helmholtz equations. *Mathematics* **2022**, *10*, 14. [CrossRef]
31. Ren, H.P.; Cheng, Y.M.; Zhang, W. Researches on the improved interpolating moving least-squares method. *Chin. J. Eng. Math.* **2010**, *27*, 1021–1029.
32. Ren, H.P.; Cheng, Y.M. The interpolating element-free Galerkin (IEFG) method for two-dimensional elasticity problems. *Int. J. Appl. Mech.* **2011**, *3*, 735–758. [CrossRef]
33. Liu, D.; Cheng, Y.M. The interpolating element-free Galerkin method for three-dimensional transient heat conduction problems. *Results Phys.* **2020**, *19*, 103477. [CrossRef]
34. Wu, Q.; Liu, F.B.; Cheng, Y.M. The interpolating element-free Galerkin method for three-dimensional elastoplasticity problems. *Eng. Anal. Bound. Elem.* **2020**, *115*, 156–167. [CrossRef]
35. Wu, Q.; Peng, P.P.; Cheng, Y.M. The interpolating element-free Galerkin method for elastic large deformation problems. *Sci. China Technol. Sci.* **2021**, *64*, 364–374. [CrossRef]
36. Qin, Y.X.; Li, Q.Y.; Du, H.X. Interpolating smoothed particle method for elastic axisymmetrical problem. *Int. J. Appl. Mech.* **2017**, *9*, 1750022. [CrossRef]
37. Wang, J.F.; Sun, F.X.; Cheng, Y.M. An improved interpolating element-free Galerkin method with nonsingular weight function for two-dimensional potential problems. *Chin. Phys. B* **2012**, *21*, 090204. [CrossRef]
38. Liu, F.B.; Cheng, Y.M. The improved element-free Galerkin method based on the nonsingular weight functions for elastic large deformation problems. *Int. J. Comput. Mater. Sci. Eng.* **2018**, *7*, 1850023. [CrossRef]
39. Liu, F.B.; Wu, Q.; Cheng, Y.M. A meshless method based on the nonsingular weight functions for elastoplastic large deformation problems. *Int. J. Appl. Mech.* **2019**, *11*, 1950006. [CrossRef]
40. Liu, F.B.; Cheng, Y.M. The improved element-free Galerkin method based on the nonsingular weight functions for inhomogeneous swelling of polymer gels. *Int. J. Appl. Mech.* **2018**, *10*, 1850047. [CrossRef]
41. Cheng, Y.M.; Peng, M.J.; Li, J.H. The complex variable moving least-square approximation and its application. *Acta Mech. Sin.* **2005**, *37*, 719–723.
42. Bai, F.N.; Li, D.M.; Wang, J.F.; Cheng, Y.M. An improved complex variable element-free Galerkin method for two-dimensional elasticity problems. *Chin. Phys. B* **2012**, *21*, 020204. [CrossRef]
43. Cheng, H.; Peng, M.J.; Cheng, Y.M. Analyzing wave propagation problems with the improved complex variable element-free Galerkin method. *Eng. Anal. Bound. Elem.* **2019**, *100*, 80–87. [CrossRef]
44. Lin, J.; Reutskiy, S. A cubic B-spline semi-analytical algorithm for simulation of 3D steady-state convection-diffusion-reaction problems. *Appl. Math. Comput.* **2020**, *371*, 124944. [CrossRef]
45. Mohanty, R.K.; Setia, N. A new high order compact off-step discretization for the system of 3D quasi-linear elliptic partial differential equations. *Appl. Math. Model.* **2013**, *37*, 6870–6883. [CrossRef]
46. Aziz, I.; Asif, M. Haar wavelet collocation method for three-dimensional elliptic partial differential equations. *Comput. Math. Appl.* **2017**, *73*, 2023–2034. [CrossRef]