An MP-DWR method for *h*-adaptive finite element methods

Chengyu Liu¹ and Guanghui Hu^{1,2,3}

¹Department of Mathematics, Faculty of Science and Technology, University of Macau, Macao SAR, China.

²Zhuhai UM Science & Technology Research Institute, Zhuhai, Guangdong, China.
³Guangdong-Hong Kong-Macao Joint Laboratory for Data-Driven Fluid Mechanics and Engineering Applications, University of Macau, Macao SAR, China.

March 9, 2023

Abstract

In a dual weighted residual method based on the finite element framework, the Galerkin orthogonality is an issue that prevents solving the dual equation in the same space as the one for the primal equation. In the literature, there have been two popular approaches to constructing a new space for the dual problem, i.e., refining mesh grids (*h*-approach) and raising the order of approximate polynomials (*p*-approach). In this paper, a novel approach is proposed for the purpose based on the multiple-precision technique, i.e., the construction of the new finite element space is based on the same configuration as the one for the primal equation, except for the precision in calculations. The feasibility of such a new approach is discussed in detail in the paper. In numerical experiments, the proposed approach can be realized conveniently with C++ *template*. Moreover, the new approach shows remarkable improvements in both efficiency and storage compared with the *h*-approach and the *p*-approach. It is worth mentioning that the performance of our approach is comparable with the one through a higher order interpolation (*i*-approach) in the literature. The combination of these two approaches is believed to further enhance the efficiency of the dual weighted residual method.

Keywords: finite element method; *h*-adaptive mesh method; dual-weighted residual; multiple-precision; C++ template

1 Introduction

The use of adjoint equations and duality arguments in residual-based *a posteriori* error estimation has become one of the most popular topics in the numerical analysis of partial differential equations. The idea can be traced back to the work of Babuška and Miller [2–4]. Then the techniques were systematically developed for more general situations [14–16]. Furthermore, Becker and Rannacher improved this approach into a computation-based feedback method, i.e. the dual weighted residual (DWR) method, which has the goal-orientation property [9–11,31]. Related techniques using duality arguments in post-processing and design have been studied in [18, 19, 27, 28]. Due to the capability of dealing with the quantities of interest, the DWR method has been applied in many areas, such as mechanics, physics, and chemistry [1,5,6,12,23,29,30].

Based on a variational formulation of the discrete problem, the DWR method uses duality techniques to generate *a posteriori* error estimation, which depends on the unknown exact solution of a so-called dual problem. It is noted that the dual solution can not be calculated in the same finite element space as the primal problem. Otherwise, the error estimation would be vanished due to the Galerkin orthogonality. To deal with this problem, there are some classic approaches to obtain an approximation of the dual solution, such as finer mesh approximation (*h*-approach), higher-order finite element method approximation (*p*-approach), or patch-wise higher-order interpolation approximation (*i*-approach) [6, 19]. In these classic approaches, several operations are necessary to construct the dual solution, which would cost unignorable computational resources. To overcome this, in this work, we explore the possibility of building a different finite element space for solving the dual problem through one feature of the computer, i.e., finite precision arithmetic.

A key observation is that with the same configuration (same mesh, degrees of freedom, basis functions, etc.), if we use two different floating point precision to build finite element spaces, for example, float-precision and double-precision, the two finite element spaces should be different from the computer's point of view. To understand it, we need to recall the way a real number is expressed in the computer. From IEEE standard shown in Fig. 1.1, each real number takes up 64 bits in double-precision formats, 32 bits in float-precision formats, and 16 bits in half-precision, respectively. Consequently, one number stored with different precision formats will be treated as different numbers in calculations.



Figure 1.1: IEEE754 data format.

Then we take two vectors $v_1 = (cos \frac{\pi}{3}, sin \frac{\pi}{3})$ and $v_2 = (-sin \frac{\pi}{3}, cos \frac{\pi}{3})$ as an example, it can be seen that two vectors are orthogonal with each other mathematically. Numerically, if two vectors are expressed in single-precision, the inner product (v_1, v_2) is computed as 0.00000000. However, if v_1 is in single, and v_2 is in double, then the inner product becomes -0.000000007771812, which is obviously not zero in double-precision. Such an observation motivates an idea for the new implementation of DWR, i.e., solving primal and dual problems in two different precision formats to avoid the Galerkin orthogonality.

In this paper, a multiple-precision DWR (MP-DWR) method is proposed for the implementation of *h*-adaptive finite element methods, towards improving the simulation efficiency. In the framework of the MP-DWR method, the primal and dual problem are solved with different precision, respectively. Based on these, a DWR *a posteriori* error estimation can be generated to serve the following local mesh refinement. The feasibility of above algorithm is discussed in detail in the paper, from which the break of the Galerkin orthogonality can be seen clearly in examples.

The improvement for the efficiency can be expected from our method by observations that i). both the re-partitioning of mesh grids and the rearrangement of degree of freedoms as well as associate basis functions are avoided in our method, and ii). the involvement of single-precision calculation would bring the reduction of both CPU time and storage. In numerical experiments, the significant acceleration by our method can be seen clearly, e.g., in solving 3D Poisson equation, around an order of magnitude difference of CPU time between the *h*-approach strategy and our method can be observed in the generation of the error indicator.

Another feature of our MP-DWR method is its concise coding work with the aid of *template* property in C++ language. In this paper, all simulations are realized by using a C++ library *AFEPack* [24], in which the code is organized following a standard procedure on solving partial differential equations as modules. By using these modules and setting the type of the floating-point number as a parameter of the template class, the code can be concise and easily managed.

The outline of this paper is as follows. In Section 2, based on the Poisson equation, the idea of goal-orientation and the dual weighted residual method in *h*-adaptive finite element methods are briefly reviewed. In Section 3, we describe a novel DWR implementation based on multiple-precision in detail. In Section 4, several numerical experiments are implemented, in which the validity and acceleration of the proposed method can be demonstrated. Besides that, the capability of the proposed method for solving the eigenvalue problem is also shown. In addition, several remarks are delivered about the properties and limitations of the proposed method, based on our numerical experience. Finally, we give the conclusion of this paper and future works in Section 5.

2 Poisson equation and DWR based *h*-adaptive mesh methods

In this section, based on the Poisson equation, a DWR based *h*-adaptive finite element method is reviewed briefly, as well as the associated classic theory.

2.1 Poisson equation and finite element discretization

Consider the Poisson equation

$$\begin{cases} -\Delta u = f, & \text{in } \Omega, \\ u = 0, & \text{on } \partial \Omega, \end{cases}$$
(1)

with a homogeneous Dirichlet boundary condition on the boundary $\partial\Omega$ for the well posedness. $\Omega \subset \mathbb{R}^d$ is a bounded domain in two (d = 2) or three (d = 3) dimensional space. We choose the variational space $V = H_0^1(\Omega)$ to incorporate the homogeneous Dirichlet boundary condition. This implies $f \in L^2(\Omega)$.

Then the weak solution *u* to the Poisson equation can be characterized via the variational problem:

Find
$$u \in V$$
, such that $a(u, v) := (\nabla u, \nabla v) = (f, v), \forall v \in V.$ (2)

On a triangulation \mathcal{T} , we can introduce the finite element space $V_h \subset H_0^1(\Omega)$. In V_h , the Ritz-Galerkin approximation to (2) can be expressed as:

Find
$$u_h \in V_h$$
, such that $a(u_h, v) := (\nabla u_h, \nabla v) = (f, v), \forall v \in V_h \subset V$, (3)

where u_h is the Galerkin approximation to u.

2.2 Goal-orientation and dual weighted residual

We now introduce the classic residual type *a posteriori* error estimation. Based on that, an introduction of the framework of a goal-oriented *a posteriori* error estimation based on DWR method follows. Denote the exact solution of (1) and (3) by u and u_h , respectively. Then the error e_h is related to the residual as:

$$a(e_h, v) = a(u - u_h, v) := \mathcal{R}(v), \quad \forall v \in V.$$

$$(4)$$

Then integration by parts element-wise to transform \mathcal{R} as:

$$\mathcal{R}(v) = \sum_{K} \int_{K} (f + \Delta u_{h}) v d\mathbf{x} + \sum_{E} \int_{E} r[\mathbf{n}_{E} \cdot \nabla u_{h}] v ds,$$
(5)

where *K* and *E* represent elements and the edges of elements, respectively, n_E represents a unit vector on *E*, and *r* is the "jump" term as defined in [33]. Based on (5), a classic residual-based error indicator η_K^{resi} for every element $K \in \mathcal{T}$ can be obtained as given in [33].

Generally, the quantities of interest can be some functionals of the exact solution u in simulations. The dual weighted residual method is a better choice to deal with these situations, in which the residual terms are multiplied by weights obtained by solving a dual problem. Following by [6,31], the framework of the DWR method can be briefly summarized as follows. Based on the above finite element discretization, to evaluate the target functional $J(\cdot)$ (assumed to be linear for simplicity), the corresponding dual problem is introduced as

$$a(\varphi, z) = J(\varphi), \ \forall \varphi \in V, \tag{6}$$

where $z \in V$ is the corresponding dual solution. The Galerkin approximation of the dual problem in V_h can be given by

$$a(\varphi_h, z_h) = J(\varphi_h), \ \forall \varphi_h \in V_h, \tag{7}$$

where $z_h \in V_h$. By combining (6) and (7), we can get the dual weighted residual error indicator of error in functional $J(\cdot)$:

$$J(e): = J(u) - J(u_h) = a(u - u_h, z) = \sum_{K \in \mathcal{T}_h} \{ (R_h, z)_K + \frac{1}{2} (r(\mathbf{n} \cdot \nabla u_h), z)_{\partial K} \} =: \eta,$$
(8)

where the cell residual R_h is defined by

$$R_{h|K} := f + \Delta u_h,\tag{9}$$

and $r(\mathbf{n} \cdot \nabla u_h)$ represents the jump term of neighboring elements *K* and *K'* on the common edge Γ with normal unit vector \mathbf{n} pointing from *K* to *K'*:

$$r(\boldsymbol{n} \cdot \nabla u_h) = (\nabla u_{h|K' \cap \Gamma} - \nabla u_{h|K \cap \Gamma}) \cdot \boldsymbol{n}.$$
(10)

In simulations, with the approximation $\tilde{\eta}$ to η , the effectivity of the error indicator can be evaluated through the index given by

$$I_{eff} = \left| \frac{\tilde{\eta}}{J(e)} \right|, \tag{11}$$

which represents the degree of overestimation and should desirably be close to one [6].

Moreover, a posterori error estimate can be obtained as

$$|J(e)| \le \eta_{\omega} := \sum_{K \in \mathcal{T}_h} \rho_K \omega_K,\tag{12}$$

where the cell residuals ρ_K and weights ω_K are given by:

$$\rho_{K} := (\|R_{h}\|_{K}^{2} + h_{K}^{-1}\|r(\boldsymbol{n} \cdot \nabla u_{h})\|_{\partial K}^{2})^{1/2},
\omega_{K} := (\|z\|_{K}^{2} + h_{K}\|z\|_{\partial K}^{2})^{1/2}.$$
(13)

Since the dual solution z is generally unknown, the only left issue is to obtain the approximation \tilde{z} . It should be noticed that \tilde{z} can not be computed in the same space V_h as the one for solving the primal problem otherwise the error indicator would be a trivial value due to Galerkin orthogonality:

$$u(u - u_h, v_h) = 0, \ \forall v_h \in V_h.$$

$$\tag{14}$$

To deal with this issue, there are three main classic approaches for computing \tilde{z} as follows.

- By *h*-approach: A denser mesh *T*_h can be constructed by globally refining the mesh *T*_h. Based on *T*_h, a different space *V*_h can be built. Then the dual solution *z* can be obtained from *V*_h.
- By *p*-approach: A finite element space $\tilde{V}_h^{(2)}$ with quadratic (or higher-order) finite elements has to be constructed. From $\tilde{V}_h^{(2)}$, the dual solution \tilde{z} can be obtained.
- By *i*-approach: The bilinear Ritz projection $z_h \in V_h$ of z is computed first. Then the approximation \tilde{z} is obtained by patch-wise higher-order interpolation of z_h . In the following numerical experiments, we only consider *quadratic* interpolation.

In practical simulations, the first two approaches all build a different finite element space for solving the dual problem. Some operations would be necessary in such a process, i.e., re-meshing the grid, re-organizing the degree of freedoms, regeneration of the information of numerical quadrature, the reformation of a larger scale stiff matrix, and solving an even larger linear system of linear equations, which would cost nonignorable CPU time and memory storage in calculations. To avoid these operations, the *i*-approach obtains the approximation \tilde{z} by using higher-order interpolation, which makes it cheaper than the other two classic approaches [29]. Different from three classic approaches above, we expect to propose a novel approach to avoid the Galerkin orthogonality. Such a new approach can also avoid these operations so the acceleration in simulations can be expected.

A novel DWR implementation based on multiple-precision 3

Recall the example in the introduction, which shows that the orthogonality of two vectors can not be preserved numerically with multiple-precision calculations. Based on the same idea, we introduce a new implementation of DWR, in which the multiple-precision will be used to effectively break the Galerkin orthogonality.

First of all, we will show how the multiple-precision breaks the Galerkin orthogonality to make sure this idea works. The Galerkin orthogonality in (14) indicates that: for $v_h = u_h$ from V_h , we have

$$a(u - u_h, u_h) = 0. (15)$$

By using a different precision format, a different space V'_h can be built without changing the mesh grid. From V'_h , u'_h can be obtained. Based on the above idea, u_h and u'_h would be different since they are computed with different precision formats. Furthermore, they are obtained from two different finite element spaces V_h and V'_h , respectively. Thus, it can be expected that the Galerkin orthogonality between them would not be preserved:

$$a(u-u_h,u_h')\neq 0. \tag{16}$$

To confirm that, we implement the numerical experiment by using a combination of double- and float-precision. Here, we only focus on the Poisson equation in the 2-dimensional domain $\Omega = (-1,1)^2$ with the exact function as $u = sin(\pi x)sin(2\pi y)$. With double-precision, the finite element space V_h^d is built firstly, from which u_h^d can be obtained. Then the similar operations are implemented in float precision, we can obtain the numerical solution u_h^f from V_h^f . Based on these, we can compute the value of $a(u - u_h^d, u_h^d)$ and $a(u - u_h^d, u_h^f)$. The numerical results are shown in Table 3.1.

Dofs	L2 Error	$a(u-u_h^d, u_h^d)$	$a(u-u_h^d, u_h^f)$
385	4.470e-1	-5.035e-5	-8.117e-5
1473	1.136e-1	-1.579e-7	-5.103e-4
5761	2.854e-2	-6.596e-10	-2.202e-3
22785	7.143e-3	8.254e-11	-9.337e-3

Table 3.1: Galerkin orthogonality of u_h^d and u_h^f for the case $u = sin(\pi x)sin(2\pi y)$.

Tables 3.1 shows that the error in L2-norm (L2 Error) tends to zero with more Dofs (degrees of freedom), such that u_h^d and u_h^f get more accurate. Meanwhile, the value of $a(u - u_h^d, u_h^d)$ tends to zero, which means the Galerkin orthogonality of u_h^d is numerically preserved better. That is the reason that we can not directly compute the dual solution in the same space with the same precision. On the contrary, the value of $a(u - u_h^d, u_h^f)$ gets further away from zero as the number of Dofs increases, which means the Galerkin orthogonality between u_h^d and u_h^f is numerically broken more obviously. Based on this phenomenon, we can confirm that the Galerkin orthogonality between u_h^d and u_h^f can be numerically broken by multiple-precision.

Thus, it can be expected that dual solutions from the finite element space constructed with different precision will work in the DWR method. By combining such primal and dual solutions, a reliable DWR error indicator can be obtained for mesh adaptation. Based on this, we propose Algorithm 1 for the multiple-precision implementation of the DWR method.

Algorithm 1 Framework of Adaptive finite element method based on MP-DWR(DF) method.

Require: The initial mesh, T_0 ; The tolerance, *tol*; The maximum iteration times, *Max*_{iter}; the iteration times k = 0.

- 1: Build a finite element space V_h on the mesh T_k ;
- 2: Solve the primal problem in double-precision to get the primal solution u_h^d ;
- 3: Solve the dual problem in float-precision to get the dual solution z_h^f ;
- 4: Calculate the cell residual $\tilde{\rho}_K$ and weights $\tilde{\omega}_K$ for each element to construct the error indicator $\tilde{\eta}_{\omega}$ as (12);
- 5: Check the stopping condition: If $\tilde{\eta} < tol$ or $k > Max_{iter}$, stop the adaptation and return the numerical solutions. 6: Else, adapt the current mesh with $\tilde{\eta}_{\omega}$ to get the new mesh \mathcal{T}_{k+1} , update the iteration times k = k + 1 and return to step 1.

Remark 1 In the above framework of MP-DWR(**DF**) method, the primal problem is solved in **D**ouble-precision. As for dual problem, it is solved with Float-precision. In fact, the framework of MP-DWR(FD) is also useful, i.e. solving the primal and dual problem in Float and Double precision, respectively. But there are some limits of MP-DWR(FD), which will be shown through a numerical example (4.4) in the following part. In this paper, MP-DWR(DF) framework is used by default in numerical experiments if not specified.

Such a new implementation of DWR method has a series of advantages, which can be summarized as follows. First of all, several repeated operations of building finite element space can be avoided. Compared to the two classic methods (*h*-approach and *p*-approach), the MP-DWR method only computes the modules of building the finite element space once in one iteration. With the help of multiple-precision, the re-partitioning of the mesh grid, the rearrangement of degrees of freedom, the recalculation of the quadrature information, the reformation of stiff matrix, and solving an even larger linear system of linear equations can be avoided in the process of obtaining the dual solution. In this way, considerable improvements in efficiency can be expected. Compared with the *i*-approach, the dual solution in MP-DWR is generally not as accurate as the one from higher-order interpolation, making the error indicator potentially less accurate. But the MP-DWR method can avoid high-order interpolation, thus saving some computation time.

The second advantage of MP-DWR comes from the involvement of computations with low precision. On computers, if one number is stored in double-precision, it will cost 8 bytes. While it costs 4 bytes in single-precision. Thus, if data are stored in single-precision instead of double-precision, the storage memory would be saved by fifty percent. Besides that, data with lower precision are more conducive to accelerating the calculation. Consider following C++ codes:

```
double double_data = 0.;
for(int i=0;i<1.0e7;i++){double_data+=4.0*atan(1.0);}
float float_data = 0.;
for(int i=0;i<1.0e7;i++){float_data+=float(4.0*atan(1.0));}</pre>
```

The same operation of the data is implemented 1.0e7 times with different precision formats. For double-precision, the cost of CPU time is 4.010e-2. As a comparison, CPU time is 2.448e-2 for float-precision, which is much less than double-precision. Therefore, it can be expected that the single-precision calculation part in the MP-DWR method can bring further reduction in both storage and CPU time.

In addition, the realization of the MP-DWR method can be very concise by using *template*. In this paper, we implement all simulations based on the *AFEPack* library [24]. The code in the *AFEPack* library follows a standard procedure for solving partial differential equations. Furthermore, each step is realized as a module based on *template* in *AFEPack*. *Template* is a feature of the C++ programming language that allows functions and classes to work on many different data types without rewriting the code for each one. By setting the floating type as the parameter of *template*, it is quite trivial to realize Algorithm 1. For example, consider the bilinear operator module in *AFEPack*: BilinearOperator < ..., Number >. We can merely change the *template* parameter 'Number', then the precision of the numbers in *BilinearOperator* will be changed. Thus, it is convenient to implement multiple-precision calculations without repeating the entire code for each precision format. Moreover, the use of *template* also facilitates the modification of our codes based on some function libraries. Codes of one numerical example of the MP-DWR method can be found in Github [25].

4 Numerical experiments

In this section, a series of examples are shown to demonstrate the validity and acceleration of the MP-DWR method. In the following numerical experiments, we use the linear triangle template element for 2D cases and the tetrahedron template element for 3D cases, respectively. All numerical examples presented in this paper are developed by using the *AFEPack* [24] in C++, and the hardware is a DELL Precision T5610 workstation with 12 CPU Cores, 3.6GHz and 64 Gigabytes memory.

Moreover, a high order Gaussian quadrature is recommended in the calculations since it can produce more accurate results, which can break the Galerkin orthogonality more obviously. Therefore, the MP-DWR error indicators can be more effective. Here, we use the six-order Gaussian quadrature in all computations.

4.1 Effectiveness of MP-DWR method

In this part, we conduct a series of numerical experiments based on MP-DWR method to verify its effectiveness. In the beginning, the Poisson equation is considered for testing the goal-orientation property and effectivity of the MP-DWR method. Then experiments of a problem with diffusion coefficient and the convection-diffusion equation are also conducted to study whether the MP-DWR method can be effective in some more complex situations.

4.1.1 Poisson equation

In this example, we consider the Poisson problem as (1) with $\Omega = (-1, 1) \times (-1, 1)$. First of all, we study the effectivity indices of the MP-DWR method for different target functionals in the Poisson case. As a benchmark example in [6], we consider an exact solution $u_1 = (1 - x_1^2)(1 - x_2^2)\sin(4x_1)\sin(4x_2)$ with the target functionals:

$$J_{\text{area}}(u) := |S|^{-1} \int_{S} u dx, \ S := [-\frac{1}{2}, 0] \times [0, \frac{1}{2}], \ x = (x_{1}, x_{2}),$$

$$J_{p}(u) := u(x_{e}), \ x_{e} = (\frac{1}{2}, \frac{1}{2}).$$
(17)

To evaluate the error indicator generated by MP-DWR, we compute the effectivity index I_{eff} given in (11). In Figure 4.1.1, we present effect indices of the MP-DWR approach applied to this Poisson case. It can be seen that effectivity indices of target functionals J_1 and J_p get closer to one as the number of Dofs increases. Although they are not strictly equal to one, they tend to around one. Therefore, we can confirm the effectivity of the MP-DWR method.



Figure 4.1.1: Effectivity indices of MP-DWR error indicators for the target functionals J_{area} and J_p over the number of Dofs.



Figure 4.1.2: Upper left: the adapted mesh based on MP-DWR. Lower left: the adapted mesh based on residual-based method. Middle: the relationship between error in target functionals and degrees of freedom. Right: the target error over the number of Dofs.

Then, a sharper function is considered, of which the exact form is given as $u_2 = 5.0 \times (1 - x_1^2)(1 - x_2^2)exp(1 - x_2^{-4})$. Besides that, we consider a point-wise output functional, which is given by:

$$J_p(u) := u(x_e), \ x_e = (0.2, 0.8).$$
(18)

And the residual-based method with indicator η_K^{resi} as mentioned above is implemented for comparison. The refined mesh and numerical results are shown in Figure 4.1.2.

From the left two pictures in Figure 4.1.2, the adapted mesh based on residual based error indicator gets denser around the singularity of function *u* in the whole domain. By contrast, the adapted mesh based on the MP-DWR method not only gets denser around the singularity but also adds much more degrees of freedom around the interested point.

Moreover, the middle picture in Figure 4.1.2 shows that the MP-DWR can obtain the numerical result reaching at the specified level with much more fewer degrees of freedom compared to the residual-based method. These phenomena confirm the goal-orientation property of the MP-DWR method, which demonstrates the effectiveness of the MP-DWR method on the Poisson problem with some particular target functionals somehow.

Furthermore, we also implement the *h*-approach, *p*-approach and *i*-approach for comparisons. The numerical results are shown in the right picture of Figure 4.1.2. It can be observed that the MP-DWR method can obtain a result of similar accuracy with a similar number of Dofs compared to the three classic approaches. Consequently, the efficiency of the MP-DWR method is comparable with the three approaches. However, the implementation of the MP-DWR method is faster, which makes the MP-DWR competitive. Such an advantage of the MP-DWR method will be shown in the following numerical example 4.2.

4.1.2 Coefficient elliptic equation

Then we consider a coefficient elliptic equation, which is given by

$$-\nabla \cdot (a\nabla u_3) + cu_3 = f, \text{ in } \Omega,$$

$$u_3|_{\partial\Omega} = 0,$$
 (19)

where

$$a = \begin{cases} 10, \ x_1^2 - x_2^2 \ge 0, \\ 1, \ x_1^2 - x_2^2 < 0. \end{cases}$$
(20)

And we set c = 1 and f = 1. The output functional of this case is defined by

$$J_p(u) := u(x_e), \ x_e = (0.5, 0).$$
⁽²¹⁾



Figure 4.1.3: Upper left: the adapted mesh based on MP-DWR. Lower left: the adapted mesh based on residual-based method. Right: the log value of target functionals.

The bilinear form of this equation should also be changed. More details can be found in [24]. Based on that, an error indicator similar to (12) can be obtained. We implement the MP-DWR method and residual-based method to refine the mesh grid, respectively. The right picture in Figure 4.1.3 shows that target error over the number of Dofs. Since the mesh is too coarse in the beginning, slight oscillation can be observed. However, with the refinement of the mesh grid, the convergence becomes smooth. Then it can be observed that the MP-DWR method can obtain the numerical result reaching the specified level with much fewer Dofs than the residual-based method. Besides that, from two left pictures in Figure 4.1.3, we can observe that the refined mesh based on MP-DWR can capture the goal point better. While the residual-based method provides a mesh, on which the solution can be approximated well in the whole domain. As a result, it has to cost more expensive numerical effort than the MP-DWR method to get accurate quantities of interest. Based on these, the effectiveness of the MP-DWR method in the coefficient elliptic equation can be verified.

4.1.3 Convection-diffusion equation

To test the validity of MP-DWR method in other complex cases, we consider the convection-diffusion equation, which is given by

$$\begin{cases} -\nabla \cdot (\epsilon \nabla u) + \alpha \nabla u = f, & \text{in } \Omega, \\ u = 0, & \text{on } \partial \Omega, \end{cases}$$
(22)

and the exact solution *u* is given as $u_4(x_1, x_2) = sin(2\pi x_1)sin(\pi x_2)$. In the numerical simulation, the ϵ is set as 0.01 and $\alpha = [1, 1]$. Following the DWR framework in the convection-diffusion equation, we can get the corresponding DWR error indicator for mesh adaptation. More details can be found in [6, 12]. For comparison, residual-based method with η_K^{resi} is implemented in the simulation. In this case, the target functional is defined as the point-wise one:

$$J_p(u) := u(x_e), \ x_e = (-\frac{1}{3}, -\frac{1}{2}).$$
(23)

Numerical results are shown in the Figure 4.1.4. This case nicely shows the goal-orientation property of MP-DWR method. It can be seen that the MP-DWR method sets more degrees of freedom around the interested point and along those finite elements that influence the error of interested point in the direction of the flow field α . As a comparison, the residual-based method puts more degrees of freedom on the whole domain. Consequently, the adapted mesh grids would be quite dense as the lower left picture in Figure 4.1.4. And the right picture in Figure 4.1.4 demonstrates that the residual-based method has to use many degrees of freedom to get an accurate numerical result. While the MP-DWR method can obtain the numerical result satisfying the same precision with just a few degrees of freedom. Therefore, we can confirm the effectiveness of MP-DWR method in this convection-diffusion case.

4.2 **Performance on acceleration**

In this part, we will show one important advantage of the MP-DWR method in numerical simulations, i.e. the acceleration of simulations. Three classic DWR approaches, i.e. *h*-approach, *p*-approach, and *i*-approach, are implemented during the experiments as comparisons. By comparing the CPU time consumption of each approach, we can test the acceleration capability of the MP-DWR method.



Figure 4.1.4: Upper left: the adapted mesh based on the MP-DWR method. Lower left: the adapted mesh based on the residual-based method. Right: the relationship between error in target functionals and degrees of freedom.

4.2.1 2D case

Above all, we consider the Poisson case as (1) with point-wise target functional J_p in (17). The exact solution is $u = 5.0 \times (1 - x_1^2)(1 - x_2^2)exp(1 - x_2^{-4})$ and the interested point is $x_e = (0.2, 0.8)$. In this case, the DWR error indicators based on four different approaches (MP-DWR and three classic approaches) are generated from the same initial mesh with almost thirty thousands degrees of freedom. The process is repeated 100 times and the average CPU time of several operations in this process is recorded in Table 4.2.1.

It can be seen that all approaches cost similar CPU time to obtain the primal solution. However, in order to generate the error indicator (including solving the dual problem and constructing the error indicator), four approaches cost quite different CPU time. Compared with *h*- and *p*-approach, the *i*-approach and MP-DWR method can save CPU time by avoiding several operations of building new space. Moreover, since no more Dofs or mesh grid points are introduced, they cost much less time to build quadrature information and solve the linear system than the former two classic approaches. In addition, the *i*-approach has to spend some time on interpolation while constructing the error indicator. Due to such a difference, the MP-DWR method is slightly faster than the *i*-approach.

Operation	<i>h</i> -approach	<i>p</i> -approach	<i>i</i> -approach	MP-DWR
Build primal FEM space	8.915e0	8.274e0	7.825e0	8.154e0
Build Quadrature information	2.632e1	2.658e1	2.255e1	2.196e1
Solve Linear System	1.788e2	1.789e2	1.836e2	1.786e2
Remeshing	1.041e1			
Build dual FEM space	3.476e1	1.092e1		
Assemble Linear System	8.231e1	4.912e1	1.837e1	1.832e1
Solve Linear System	7.039e0	2.669e1	1.238e0	1.203e0
Construct indicator	2.999e1	2.934e1	3.420e1	2.815e1
Generate indicator	1.645e2	1.161e2	5.381e1	4.767e1
Total CPU time	3.785e2	3.298e2	2.678e2	2.564e2

Table 4.2.1: CPU time(seconds) of operations in the computation of Poisson case. The values are the average of 100 experimental results. The process of generating the indicator includes solving the dual problem and constructing the error indicator. For the *i*-approach, the CPU time for interpolation is included in the CPU time for the 'construct indicator' step.

Then we implement the whole numerical simulation through each approach. The mesh adaptation is executed until the error tolerance is satisfied. The numerical results are shown in the left one of Figure 4.2.1. Furthermore, we change the exact solution of Poisson case as $u_1 = (1 - x_1^2)(1 - x_2^2) \sin(4x_1) \sin(4x_2)$ with the area target functional $J_{area}(u)$ in (17). Similar simulations are implemented. The relationship between the error and whole CPU time for this case is shown in the right one of Figure 4.2.1. In the left picture in Figure 4.2.1, it can be seen that the MP-DWR method and *i*-approach are much faster than *h*- and *p*-approach. The acceleration gets more obvious as the times of mesh adaptation increase. Then we focus on the comparison of MP-DWR and *i*-approach: the *i*-approach can control the error of the point-wise target functional better than MP-DWR method with a similar number of Dofs. But the CPU time of the MP-DWR method is always less than the *i*-approach. This suggests that the MP-DWR method may be less accurate but computationally fast compared to the *i*-method in this case.



Figure 4.2.1: The error and CPU time of the whole experiments based on MP-DWR and three classic DWR approaches. Left: with point-wise target functional. Right: with area target functional. The number of Dofs is shown in the figure.

As for the right picture, the corresponding points of the MP-DWR method always stay at the most left position in each iteration step, which means that the MP-DWR method generally cost the least CPU time in simulations. When the iterations stop, although the MP-DWR method uses the most Dofs, the CPU time of the MP-DWR method is significantly less than the other three methods. This demonstrates that the MP-DWR method can be calculated faster than the three classical methods in this case.

4.2.2 3D case

Then we consider the Poisson equation in 3-dimensional with a target functional, which is given by:

$$J_G(u) = \int_{\Omega_{3D}} u dx, \ x = (x_1, x_2, x_3), \tag{24}$$

where $\Omega_{3D} = (-1, 1)^3$. The exact solution is given as the example in [6]:

$$u = (1 - x_1^2)^2 (1 - x_2^2)^2 (1 - x_3^2)^2 (64x_1^2 + 0.1)^{-1} (64x_3^2 + 0.1)^{-1}.$$
(25)

The comparison of CPU time for the whole simulation based on the MP-DWR method and three classic approaches is similar to the 2D case in Figure 4.2.1. The MP-DWR method and *i*-approach demonstrate similar efficiency, and both of them are computed faster than the *h*- and *p*-approach.

Then we mainly focus on the comparison of CPU time for generating the error indicator. Similar to the 2D case, we implement the four approaches to generate the error indicators from the same initial mesh (with almost 18000 Dofs), respectively. The corresponding CPU time is recorded in the Table 4.2.2. The values are the average of 100 experimental results.

Compared to the *i*-approach, the MP-DWR method can save some CPU time during solving the dual problem since the involvement of single-precision computations. Besides that, due to the increased number of elements and corresponding neighbors in the 3D case, higher-order interpolation takes more CPU time than in the 2D case. Consequently, the MP-DWR method can save much CPU time during constructing the indicator. More importantly, compared to the *h*- and *p*-approach, the improvement in CPU time brought by the MP-DWR method is quite significant. The MP-DWR method can use the equivalent of about 15% of the time for the *h*-approach and about 30% of the time for the *p*-approach to generate the indicator. Based on that, we can demonstrate that the performance of the MP-DWR method in acceleration is quite considerable in the 3D case compared to the *h*- and *p*-approach. Therefore, the MP-DWR method can be more competitive in 3D numerical simulations.

4.3 Performance in eigenvalue problems

To extend the MP-DWR method into more applications, we spend some time on verifying its effectiveness in the eigenvalue problem. Consider the eigenvalue problem as follows. Seeking an eigenpair $(\lambda, u) \in C \times H_0^1(\Omega)$ with

$$-\Delta u + \boldsymbol{\beta} \cdot \nabla u = \lambda u \text{ in } \Omega. \tag{26}$$

In experiments, β is set as [3,0]. The computational domain is set as the slit area $\Omega_3 = ((-1, 1) \times (-1, 1)) \setminus ([0, 1] \times \{0\})$. Following the DWR framework in the eigenvalue problem as in [6,17,20], we can get the related error indicator for mesh adaptation. In the simulation of the MP-DWR method, we follow the framework of Algorithm 1, i.e. the primal solution and corresponding residual ρ_K are computed in double-precision. While the dual solution and corresponding residual ρ_K^* are computed in single-precision. Based on these, the error indicator $\eta_{\lambda}^{\omega} = \sum_{K \in \mathcal{T}_h} \{\rho_K \omega_K^* + \rho_K^* \omega_K\}$ can be computed. For comparison, the residual-based error indicator $\eta_{\lambda}^{\text{energy}} = c_{\lambda} \sum_{K \in \mathcal{T}_h} h_K^2 \{\rho_K^2 + \rho_K^{*2}\}$ is also implemented, where c_{λ} is a

Operation	<i>h</i> -approach	<i>p</i> -approach	<i>i</i> -approach	MP-DWR
Build primal FEM space	3.086e0	3.048e0	2.825e0	3.196e0
Build Quadrature information	1.135e1	1.137e1	1.129e1	9.710e0
Solve Linear System	2.137e1	2.137e1	2.139e1	2.148e1
Remeshing	6.423e0			
Build dual FEM space	2.446e1	3.886e0		
Assemble Linear System	7.833e1	4.096e1	9.853e0	8.272e0
Solve Linear System	1.062e0	6.402e0	1.449e-1	1.447e-1
Construct indicator	9.956e0	1.033e1	2.157e1	9.691e0
Generate indicator	1.202e2	6.162e1	3.156e1	1.811e1
Total time	1.560e2	9.741e1	6.707e1	5.249e1

Table 4.2.2: CPU time(seconds) of operations in 3-dimensional Poisson case. The values are the average of 100 experimental results. The process of generating the indicator includes solving the dual problem and constructing the error indicator. For the *i*-approach, the CPU time for interpolation is included in the CPU time for the 'construct indicator' step.



Figure 4.3.1: Upper left: the plot figure of the primal solution. Lower left: dual solution.. Right: the smallest eigenvalue errors over the number of Dofs.

constant growing linearly with $|\lambda|$. The left pictures in Figure 4.3.1 shows the numerical primal and dual solution. The residual of the smallest eigenvalue λ_1 over Dofs is shown in the right one of Figure 4.3.1. It can be observed that, with the specified number of Dofs, the MP-DWR method can get a more accurate result than the η^{energy} based method. This demonstrates that the MP-DWR method can deal with the interested eigenpair more efficiently. Consequently, it can be demonstrated that the MP-DWR method can still work in the eigenvalue problem. Based on these, it can be expected that the MP-DWR method can be used in many applications, such as density functional theory.

4.4 The limit of MP-DWR method

In this part, we discuss a limit of the MP-DWR(FD) framework. In some numerical experiments, it is strange that MP-DWR(FD) becomes less effective with refining the mesh too many times. Consider the Poisson case with point-wise target functional in case 4.1.1. We solve the primal problem with single-precision and double-precision, respectively. The numerical results of the primal solutions are shown in Figure 4.4.1. It can be seen that when the mesh grids are dense enough, the residual in single-precision gets larger as the number of degrees of freedom increases, which is different from the residual in double-precision. With this unexpected phenomenon, it can not be expected that such an inaccurate primal solution in single-precision can construct a reliable DWR error indicator. One possible reason is that while we adapt the mesh several times, some area will gets quite dense and the minimal volume of elements will get very tiny. For reference, the minimal volumes of elements on the mesh grids in each iteration are shown in Table 4.4. When the minimal volume gets the level of 1.0e-6, the data in such an element can be close to the limit of single-precision so that the computations may demonstrate some unexpected phenomena causing the strange curve in Figure 4.4.1. Due to such a limit, we have to be careful about the data precision in numerical simulations while using the MP-DWR method.

Remark 2 Although there is a limit to the MP-DWR(FD) method, we propose a potential algorithm to make full use of the advantages of the MP-DWR method: First of all, we can compute the primal solution with half-precision and dual solution with single-precision, respectively. When the primal solution gets the limit of half-precision, we change the precision parameters in our programming to compute the primal solution with single-precision and dual solution with double-precision, respectively. In this



Figure 4.4.1: Primal solutions computed in float and double precision.

Adapt times	1	2	3	4	5
Minimal element volume	1.617e-2	1.010e-3	6.316e-05	4.34e-06	1.247e-06

Table 4.4.1: The minimal volume	e of elements in adaptation
---------------------------------	-----------------------------

way, we can make full use of the acceleration of the MP-DWR method to get a reliable numerical result. And such an algorithm can also be an efficient preconditioned approach.

5 Conclusion

To accelerate the process of obtaining dual solution in the DWR methods, we utilize the properties of precision formats and propose a novel DWR implementation based on multiple-precision named MP-DWR. Numerical experiments confirm the effectiveness of the novel approach. Memory storage savings can be expected due to the low-precision calculations involved. Furthermore, considerable reduction in CPU time compared with classic *h*- and *p*-approach is shown in simulations, which makes the MP-DWR method quite competitive. Compared with *i*-approach, the MP-DWR may lose some accuracy but it gains some acceleration in simulations, especially in the 3D case. By combining the MP-DWR method and *i*-approach, both of the accuracy and efficiency might be improved in simulations, which is our ongoing work. Although there is a limit to the MP-DWR method due to the property of precision formats in some situations, the substantial improvements in data storage and CPU time suggest that it is still an efficient method.

In addition, some more precise floating-point data type than double, such as __float128(long double), may be applied in the MP-DWR framework to overcome the limit. Such an extension of the MP-DWR method will be implemented in our future study. In our previous work, applications of the *h*-adaptive method to computational fluid dynamics, micromagnetics, and density functional theory have been studied [7, 8, 21, 26, 34]. The introduction of the MP-DWR method into these applications to further improve the efficiency is also our future work. Besides that, the dual weighted residual method has been applied to many time-dependent problems [13, 22, 32], for which the MP-DWR is expected to be used to improve the computational efficiency.

Acknowledgement

The research of G. Hu was partially supported by National Natural Science Foundation of China (Grant Nos. 11922120 and 11871489), FDCT of Macao SAR (0082/2020/A2), MYRG of University of Macau (MYRG2020-00265-FST) and Guangdong-Hong Kong-Macao Joint Laboratory for Data-Driven Fluid Mechanics and Engineering Applications (2020B1212030001).

Statements and Declarations

Data availability statements

Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Competing interests

The authors declare that they have no conflict of interest.

References

- [1] D Avijit and Srinivasan Natesan. An efficient DWR-type a posteriori error bound of SDFEM for singularly perturbed convection–diffusion PDEs. *Journal of Scientific Computing*, 90(2):1–16, 2022.
- [2] Ivo Babuška and A Miller. A feedback finite element method with a posteriori error estimation: Part I. The finite element method and some basic properties of the a posteriori error estimator. *Computer Methods in Applied Mechanics and Engineering*, 61(1):1–40, 1987.
- [3] Ivo Babuška and Anthony Miller. The post-processing approach in the finite element method—part 2: the calculation of stress intensity factors. *International Journal for numerical methods in Engineering*, 20(6):1111–1129, 1984.
- [4] Ivo Babuška and Anthony Miller. The post-processing approach in the finite element method—Part 1: calculation of displacements, stresses and other higher derivatives of the displacements. *International Journal for numerical methods in engineering*, 20(6):1085–1109, 1984.
- [5] Wolfgang Bangerth, Michael Geiger, and Rolf Rannacher. Adaptive galerkin finite element methods for the wave equation. *Computational Methods in Applied Mathematics*, 10(1):3–48, 2010.
- [6] Wolfgang Bangerth and Rolf Rannacher. Adaptive Finite Element Methods for Differential Equations.
- [7] Gang Bao, Guanghui Hu, and Di Liu. An h-adaptive finite element solver for the calculations of the electronic structures. *Journal of Computational Physics*, 231(14):4967–4979, 2012.
- [8] Gang Bao, Guanghui Hu, and Di Liu. Numerical solution of the Kohn-Sham equation by finite element methods with an adaptive mesh redistribution technique. *Journal of Scientific Computing*, 55(2):372–391, 2013.
- [9] Roland Becker and R. Rannacher. A feed-back approach to error control in finite element methods: Basic analysis and examples. *East-West Journal of Numerical Mathematics*, 4, 01 1996.
- [10] Roland Becker and Rolf Rannacher. Weighted a posteriori error control in FE methods. 1996.
- [11] Roland Becker and Rolf Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta numerica*, 10:1–102, 2001.
- [12] Marius Paul Bruchhäuser, Kristina Schwegler, and Markus Bause. Numerical study of goal-oriented error control for stabilized finite element methods. In *Chemnitz Fine Element Symposium*, pages 85–106. Springer, 2017.
- [13] Marius Paul Bruchhäuser, Kristina Schwegler, and Markus Bause. Dual weighted residual based error control for nonstationary convection-dominated equations: potential or ballast? In *Boundary and Interior Layers, Computational* and Asymptotic Methods BAIL 2018, pages 1–17. Springer, 2020.
- [14] Kenneth Eriksson, Don Estep, Peter Hansbo, and Claes Johnson. Introduction to adaptive methods for differential equations. *Acta numerica*, 4:105–158, 1995.
- [15] Kenneth Eriksson and Claes Johnson. An adaptive finite element method for linear elliptic problems. *Mathematics of Computation*, 50(182):361–383, 1988.
- [16] Kenneth Eriksson and Claes Johnson. Adaptive finite element methods for parabolic problems I: A linear model problem. *SIAM Journal on Numerical Analysis*, 28(1):43–77, 1991.
- [17] Joscha Micha Gedicke. *On the numerical analysis of eigenvalue problems*. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, 2013.
- [18] Michael Giles, M Larson, M Levenstam, and Endre Suli. Adaptive error control for finite element approximations of the lift and drag coefficients in viscous flow. 1997.
- [19] Michael B Giles and Endre Süli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. *Acta numerica*, 11:145–236, 2002.
- [20] Vincent Heuveline and Rolf Rannacher. A posteriori error control for finite element approximations of elliptic eigenvalue problems. *Advances in Computational Mathematics*, 15(1):107–138, 2001.
- [21] Guanghui Hu, Xucheng Meng, and Nianyu Yi. Adjoint-based an adaptive finite volume method for steady euler equations with non-oscillatory k-exact reconstruction. *Computers & Fluids*, 139:174–183, 2016.
- [22] Uwe Köcher, Marius Paul Bruchhäuser, and Markus Bause. Efficient and scalable data structures and algorithms for goal-oriented adaptivity of space–time fem codes. *SoftwareX*, 10:100239, 2019.
- [23] Katharina Kormann. A time-space adaptive method for the Schrödinger Equation. *Communications in Computational Physics*, 20(1):60–85, 2016.
- [24] Ruo Li. On multi-mesh h-adaptive methods. Journal of Scientific Computing, 24(3):321–341, 2005.
- [25] Chengyu Liu and Gunaghui Hu. https://github.com/Mixed-DWR/MP-DWR_Example2.git.
- [26] Xucheng Meng and Guanghui Hu. A nurbs-enhanced finite volume method for steady euler equations with goaloriented *h*-adaptivity. *Communications in Computational Physics*, 2022.
- [27] J Peraire and AT Patera. Bounds for linear-functional outputs of coercive partial differential equations: local indicators and adaptive refinement. *Studies in Applied Mechanics*, 47:199–216, 1998.
- [28] Serge Prudhomme and J Tinsley Oden. On goal-oriented error estimation for elliptic problems: application to the control of pointwise errors. *Computer Methods in Applied Mechanics and Engineering*, 176(1-4):313–331, 1999.

- [29] Ehsan Rabizadeh, Amir Saboor Bagherzadeh, Cosmin Anitescu, Naif Alajlan, and Timon Rabczuk. Pointwise dual weighted residual based goal-oriented a posteriori error estimation and adaptive mesh refinement in 2d/3d thermomechanical multifield problems. *Computer Methods in Applied Mechanics and Engineering*, 359:112666, 2020.
- [30] R. Rannacher. Error Control in Finite Element Computations. Springer Netherlands, Dordrecht, 1999.
- [31] Rolf Rannacher. Adaptive galerkin finite element methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128(1-2):205–233, 2001.
- [32] Michael K. Sleeman and Masayuki Yano. Goal-oriented model reduction for parametrized time-dependent nonlinear partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 388:114206, 2022.
- [33] Rüdiger Verfürth. A posteriori error estimation techniques for finite element methods.
- [34] Lei Yang and Guanghui Hu. An adaptive finite element solver for demagnetization field calculation. *Advances in Applied Mathematics and Mechanics*, 2019.