Algorithm of h-adaptation

The goal of the h adaptation algorithm is to improve the accuracy of approximation on computational meshes by breaking selected elements into smaller ones. The h plane adaptation algorithm can be implemented only when we construct such basic functions that can be spread on the elements of finished texts. In the case of unstructured meshes composed of triangular elements, it is possible to use the Rivary algorithm described in this chapter.
In the case of meshes of structural elements, built of structural elements, on which basis functions begin according to given node vectors, adaptation is also possible, if it concerns structural elements, C^0 separators.
The chapter "Approximation with hanging nodes" describes the method of joining together the base functions spanning on adjacent broken elements of various sizes.
The h-adaptation algorithm can be implemented in the form of an a priori or a posteriori algorithm, and using isotropic or anisotropic adaptations.
The adaptive algorithm can use one computational mesh, the so-called sparse mesh, or two sparse and dense computational meshes. In our classification by a priori adaptive algorithm, we mean an algorithm that makes a decision before constructing and solving a problem on a dense mesh, after solving a problem on an actual sparse mesh. For this reason, the term "a priori" as if "in advance" is understood here without
additional calculations related to, for example, problem solving on an additional dense mesh. By a posterior adaptive algorithm we mean an algorithm that makes decisions about adaptation after performing some additional calculations (not only solving the problem on a sparse mesh), for example after generating a dense mesh and solving a problem on a dense mesh.

- A priori h-adaptation algorithm uses one computational grid, and error estimation is based on local calculations on individual elements (for example, it is possible to estimate errors based on the first and / or second directional derivatives of the solution locally on finite elements). Many methods of selecting adaptations are described in Mark Ainsworth and John Tinsley Oden [1] (there is a discrepancy in the nomenclature of methods between our module and this article, resulting from the fact that we call methods and priory methods using only a sparse mesh and a posteriori methods using two meshes, in the authors the method using one mesh is called a posteriori because it is run after solving the problem, for the Authors, the a priori method would make decisions about adaptation before solving the computational problem, looking for example at the coefficients of the model and the shape of the computational grid).
- The a posteriori h-adaptation algorithm uses two computational meshes, a sparse mesh and a dense mesh, and the error estimation on individual elements is done by comparing the solutions on the sparse and dense meshes
- The h-isotropic adaptation algorithm can work in a prior iia posteriori version, and it divides selected elements into four (rectangular elements in two dimensions) or eight (cubic elements in three dimensions) new elements (various breaking of triangular elements in two dimensions, and tetrahedral, prismatic and pyramids in three dimensions; recently the finite element method on polygons has also been used, e.g. the PolyDPG method)
- Anisotropic h-adaptation algorithm can work in a priori and a posteriori versions, and it divides selected elements into two new elements in the selected direction

As prof. Zienkiewicz, one of the creators of the finite element method, we will get sufficient accuracy in many engineering problems using square polynomials and h-adaptation. So we assume that the initial mesh is a glued group of elements described by vectors of nodes 0 0 0 1 1 1 and 0 0 0 1 1 1.

The a priori anniotropic h-adaptation algorithm can be written as follows:

WŁASNOŚĆ

Własność 1: Algorithm h-adaptacji anioztropowej a priori

1. inhomogeneous h adaptations performed apropri (initial grid, accuracy, maximum number of steps)
2. current grid = initial grid
3. solve your computational problem on the current mesh (for example, bitmap projection or heat transport) by computing the current solution u
4. loop for i = 1 up to the maximum number of steps
5. $K_{maxerr} = 0; K^x_{maxerr} = 0; K^y_{maxerr} = 0$
6. for each element K the current grid, do the following
7. calculate the approximation error on the element K by computing the solution gradient semormat on the element $K_{err} = \|\nabla(B-u)\| = \int |\frac{\partial(B-u)}{\partial x}|^2 + |\frac{\partial(B-u)}{\partial y}|^2 dxdy$, and norms from directional derivatives $K^x_{err} = \|\nabla_x(B-u)\| = \int |\frac{\partial(B-u)}{\partial x}|^2 dxdy$ and $K^y_{err} = \|\nabla_y(B-u)\| = \int \|\frac{\partial(B-u)}{\partial y}|^2 dxdy$
8. if $K_{err} > K_{maxerr}$ then $K_{maxerr} = K_{err}$
9. if $K^x_{err} > K^x_{maxerr}$ then $K^x_{maxerr} = K_{err}$
10. if $K^y_{err} > K^y_{maxerr}$ then $K^y_{maxerr} = K_{err}$
11. end of the loop by elements
12. if $K_{maxerr} >$ the accuracy required is over
13. loop by elements K mesh
14. if $K_{err} > 0.33 * K_{maxerr}$ then break the element in two directions into four new elements and generate new local basis functions, continue to loop over the elements

15. if $K^x_{err} > 0.33 * K^x_{maxerr}$ then break the element in the direction of the axis x for two new elements, and generate new base functions, continue looping over the elements
16. if $K^y_{err} > 0.33 * K^y_{maxerr}$ then break the element in the direction of the axis y and generate new base functions, continue looping over the elements
17. end the loop over the elements
18. end of the adaptation loop

By an error in the example bitamp projection task, we mean an error in the norm H^1 representing the difference between a continuous approximation of, for example, a cast of terrain, the height of which at different points is described by the shades of bitmap pixels.
If it is necessary to break a given element (perform h-adaptation for this element), we have three scenarios to choose from. The first option is to break an element into four smaller pieces. The second option is to break the element into two new elements in the direction of the axis
x. The third possibility is breaking a given element into two smaller elements in the direction of the axis y.
In the adaptive algorithm, usually no adaptation is performed on all mesh elements, but only on those elements where the error is greater than 33 percent of the maximum error (calculated over all elements). This is, of course, an arbitrarily adopted value, selected on the basis of numerical experiments by prof. Leszek Demkowicz. In the proposed version of the algorithm, we first try to adapt in both directions (break the element in both directions), and then (if it is not indicated) we try to break the element in one of two directions. This is obviously an example version of the algorithm, various modifications are possible. The most advanced version of the adaptive algorithm is the hp adaptation algorithm described in another module.
Generating new base functions consists in generating new local node vectors for broken elements. For example, if we have an element described by a node vector 0 0 0 1 1 1 in the direction of the axis x and vector of nodes 0 0 0 1 1 1 in the direction of the axis y, and we break this element into four new elements, then we have a vector of nodes 0 0 0 0.5 0.5 1 1 1 in the direction of the axis x and wektor węzłów 0 0 0 0.5 0.5 1 1 1 in the direction of the axis y.
If we will break this element into two new elements in the direction of the axis x, wówczas mamy wektor węzłów 0 0 0 0.5 0.5 1 1 1 in the direction of the axis x and wektor węzłów 0 0 0 0 1 1 1 in the direction of the axis y.
If we will break this element into two new elements in the direction of the axis y, wówczas mamy wektor węzłów 0 0 0 1 1 1 in

the direction of the axis x and vector of nodes 0 0 0 0.5 0.5 1 1 1 in the direction of the axis y.

By merging many patches (groups of elements) on which different B-spline polynomials are spanned, we will obtain a computational mesh on which local modifications of the degree of the B-spline function will be possible, for example using the described p-adaptation algorithm. The method of merging the base functions obtained on large and small broken elements is described in the chapter "Approximation with hanging nodes".

Automatic aposteriori h-adaptation algorithm:

ALGORYTM

Algorytm 1: Algorithm automatycznej h-adaptacji aposteriori

Automatic h-adaptation algorithm (initial grid, required accuracy, maximum number of iterations)

1. coarse mesh = initial mesh
2. loop i = 1 up to the maximum number of iterations
3. Solve a computational problem on the current sparse mesh (for example, bitmap projection or heat transport problem) by obtaining u_h
4. coarse mesh = sparse mesh
5. Break each sparse mesh element into four elements
6. Solve the computational problem on the current dense mesh and obtain a solution

$u_{h/2}$

1. Maximum error $K_{max} = 0$
2. Loop by elements K sparse mesh
3. For each sparse mesh element, estimate the relative error (the norm of the difference between the solution on the sparse and dense mesh)

$$K_{rel} = \|u_h - u_{h/2}\|_K = \int_K \left(u_h - u_{h/2}\right)^2 + \left(\frac{\partial u_h - u_{h/2}}{\partial x}\right)^2 + \left(\frac{\partial u_h - u_{h/2}}{\partial y}\right)^2$$

1. If $K_{rel} > K_{max}$ then $K_{max} = K_{rel}$
2. End of the loop by elements

3. If K_{max} > required accuracy, then it's over.
4. Loop by elements K sparse mesh
5. If $K_{rel} > 0.33 K_{max}$ then select the optimal method of adapting the element from the sparse mesh and apply it to the element K sperse mesh
6. End of the loop through the elements
7. End of iteration

So we have three options for adapting a single element:

1. Break an element into four new elements
2. Breaking an element into two new elements in the horizontal direction
3. Breaking an element into two new elements vertically

How is the decision about the type of adaptation of a single element made?

The decision is made in accordance with the following Algorithm.

ALGORYTM

Algorytm 2: Algorithm for selecting the optimal strategy for component adaptation

1. Choosing the optimal strategy for component adaptation K (solution on a sparse mesh narrowed to an element u_h, dense mesh solution narrowed to the element $u_{h/2}$)
2. Loop through possible types of element adaptation from 1 to 3
3. The fastest error rate = 0
4. Optimal adaptation = 0
5. Make a copy J of the element K and make the considered adaptation on it
6. Calculate the projection w solutions on a dense mesh $u_{h/2}$ on the element being adapted J
7. Calculate how much the relative error will drop if we will adapt the code, the error will drop (kod)= $\|u_{h/2} - u_h\|_K - \|u_{h/2} - w\|_K$
8. Calculate how many unknowns (how many base functions) we have to add to implement the adaptation represented by the code, adaptation cost (code)
9. Calculate and remember the rate of error decrease (code) = error decrease (code) / adaptation cost (code)
10. If the error decrease rate (code) is greater than the fastest error decrease speed, then remember the largest error decrease value = error decrease rate (code), optimal adaptation = code
11. End of the loop after possible types of adaptation
12. Perform on item K the optimal adaptation found

In other words, we choose the type of adaptation for the element that achieves the highest error reduction at the lowest cost. This quantity is represented by the rate of error decrease, which increases with decreasing error but decreases with the cost incurred (with the number of functions added to the element because the cost of the calculation on a given element depends on the number of unknowns, which are coefficients of the base functions).

Bibliografia

1. Mark Ainsworth, John Tinsley Oden: A posteriori error estimation in finite element analysis, Computer Methods in Applued Mechanics and Engineering, Elsevier 1997, dostęp:29.03.2020

Utworzona przez admin. Ostatnia aktualizacja: Środa 28 z Październik, 2020 08:36:03 UTC przez paszynsk@agh.edu.pl. Autor: Maciej Paszynski