paszynsk@agh.edu.pl

Algorithm of p-adaptation

The purpose of the p-adaptation algorithm is to increase the accuracy of approximation on computational grids by increasing the degree of polynomials spanning the computational grid. The p adaptation algorithm is possible to implement only when we construct such base functions spanning on finite elements for which it is possible to raise the degree of polynomials. In the case of traditional B-spline functions, described in greater detail in this manual, it is only possible to increase the degree of the B-spline function globally on a local finite element patch.
In other words, suppose we have a node vector describing the basis of second order B-splines with C1 continuity in the direction of the axis x 0 0 0 1 2 3 4 4 4, and an identical node vector describing the base of the second order B-splines in the direction of the axis y 0 0 0 1 2 3 4 4 4.
These vectors span two one-dimensional bases of B-splines in the direction of individual axes
$B^x_{1;2}(x), \ldots, B^x_{6;2}(x); B^y_{1;2}(y), \ldots, B^y_{6;2}(y)$
which through a tensor product define the basis of two-dimensional basis functions
$\{B^x_{i;2}(x)B^y_{j;2}(y)\} i = 1, \ldots, 6; j = 1, \ldots, 6$
The following adaptations are possible:

- increasing the degree of B-spline function in the direction of the axis x o 1, by modifying the vector of nodes in the direction of the axis x 0 0 0 0 1 2 3 4 4 4 4. We will then get the bases $B^x_{1;3}(x), \ldots, B^x_{7;3}(x); B^y_{1;2}(y), \ldots, B^y_{6;2}(y)$ and two-dimensional functions $\{B^x_{i;2}(x)B^y_{j;2}(y)\} i = 1, \ldots, 7; j = 1, \ldots, 6$

- increasing the degree of B-spline function in the direction of the axis y o 1, by modifying the vector of nodes in the direction of the axis y 0 0 0 0 1 2 3 4 4 4 4. We will then get the bases $B^x_{1;3}(x), \ldots, B^x_{7;3}(x); B^y_{1;2}(y), \ldots, B^y_{6;2}(y)$ oraz dwuwymierowe funkcje $\{B^x_{i;2}(x)B^y_{j;2}(y)\} i = 1, \ldots, 6; j = 1, \ldots, 7$
- increasing the degree of the B-spline function in the direction of both axes, by modifying the vector of nodes in both directions 0 0 0 0 1 2 3 4 4 4 4 and 0 0 0 0 1 2 3 4 4 4 4 by 1. Then we will get base $B^x_{1;3}(x), \ldots, B^x_{7;3}(x); B^y_{1;3}(y), \ldots, B^y_{7;3}(y)$ and two-dimensional functions $\{B^x_{i;2}(x)B^y_{j;2}(y)\} i = 1, \ldots, 7; j = 1, \ldots, 7$

We can of course increase the degrees of the base functions in one direction or the other by any degree (as long as our computer program can generate a higher degree B-spline without any problem).
For B-spline functions, it is not possible to mix degrees of polynomials without placing {OPENAGHMATHJAX (type = "inline" anchor = "eq10: 13" isNumeration = "yes")} C ^ 0 {OPENAGHMATHJAX} separators between B-spline functions. However, it is possible to mix patches (groups of elements) separated by separators, on which B-spline polynomials have different continuity. For example, suppose we have a node vector describing the basis of second order B-splines with C1 continuity on 2 elements in the direction of the axis x and a node vector describing the basis of third order B-splines of C2 continuity on the next 2 elements in the axis direction x czyli 0 0 0 1 2 2 2 2 2 2 3 4 4 4 4.
These vectors span the base together
$B^x_{1;2}(x), \ldots, B^x_{6;2}(x); B^y_{1;2}(y), \ldots, B^y_{6;2}(y)$
We can "assemble" these vectors with the base of second degree B-splines in the direction of the axis y 0 0 0 1 2 3 4 4 4, czyli $B^x_{1;2}(x), \ldots, B^x_{6;2}(x); B^y_{1;2}(y), \ldots, B^y_{6;2}(y)$.
These vectors span the basis of the B-spline function in the direction of the axis x in the following way
$B^x_{1;2}(x), B^x_{2;2}(x), B^x_{3;2}(x), B^x_{4;2}(x), B^x_{5;3}(x), B^x_{6;3}(x), B^x_{7;3}(x), B^x_{8;3}(x), B^x_{9;3}(x)$.
Through the tensor product, we obtain the basis of two-dimensional basis functions
$\{B^x_{i;2}(x)B^y_{j;2}(y)\} i = 1, \ldots, 4; j = 1, \ldots, 6\} \cup \{B^x_{i;2}(x)B^y_{j;2}(y)\} i = 5, \ldots, 9; j = 1, \ldots, 6\}$
Note that the groups of elements glued in this way do not give a continuous approximation at point 2. On the left side we have one polynomial maxing out 1 at point 2, on the right side we have another other polynomial maxing out at 1 at point 2.
The node vector pair 0 0 0 1 2 2 2 2 2 2 3 4 4 4 4 is equivalent to the vector 0 0 0 1 2 2 2 2 3 4 4 4 4. Of course, the algorithm that interprets such a vector of nodes must notice that starting from node 2, which was repeated 4 times, the degree of polynomials is increased by 1. Additionally, the last polynomial in the first subgroup $B^x_{4;2}(x)$ należało by scalić z pierwszym wielomianem w

drugiej podgrupie elementów $B^x_{5;3}(x)$. Such a merge is possible because the first and last terms of B-splines are identical polynomials regardless of their degree. We will then obtain a second order polynomial $B^x_{4;2}(x)$ which extends to the last element in the first subgroup and the first element in the second subgroup. So our bases now look like this
$B^x_{1;2}(x), B^x_{2;2}(x), B^x_{3;2}(x), B^x_{4;2}(x), B^x_{5;3}(x), B^x_{6;3}(x), B^x_{7;3}(x), B^x_{8;3}(x).$
$B^x_{1;2}(x), \ldots, B^x_{6;2}(x); B^y_{1;2}(y), \ldots, B^y_{6;2}(y)$
$\{B^x_{i;2}(x)B^y_{j;2}(y)\} i = 1, \ldots, 4; j = 1, \ldots, 6\} \cup \{B^x_{i;2}(x)B^y_{j;2}(y)\} i = 5, \ldots, 8; j = 1, \ldots, 6\}$

ALGORYTM

Algorytm 1: Homogeneous p adaptation algorithm performed apropri

The general form of p-adaptation algorithm can be written as follows:

1. homogeneous p adaptation 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

1. loop for i = 1 up to the maximum number of steps
2. $K_{maxerr} = 0; K^x_{maxerr} = 0; K^y_{maxerr} = 0$
3. for each item K current grids do the following
4. compute approximation error on 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 $K^x_{err} = \|\nabla_x(B-u)\| = \int |\frac{\partial(B-u)}{\partial x}|^2 dxdy$ oraz
$K^y_{err} = \|\nabla_y(B-u)\| = \int \|\frac{\partial(B-u)}{\partial y}|^2 dxdy$

1. if $K_{err} > K_{maxerr}$ then $K_{maxerr} = K_{err}$
2. if $K^x_{err} > K^x_{maxerr}$ then $K^x_{maxerr} = K_{err}$
3. if $K^y_{err} > K^y_{maxerr}$ then $K^y_{maxerr} = K_{err}$

4. end of loop by elements
5. if $K_{maxerr} <$ the accuracy required is over
6. loop by elements K mesh
7. if $K_{err} > 0.33 * K_{maxerr}$ then pick up the degree of polynomial approximation in two directions, continue looping over the elements
8. if $K^x_{err} > 0.33 * K^x_{maxerr}$ then pick up the degree of polynomial approximation in the direction of the axis x, continue looping through the items
9. if $K^x_{err} > 0.33 * K^y_{maxerr}$ then pick up the degree of polynomial approximation in the direction of the axis y, continue looping through the items
10. end the loop over the elements
11. end of the adaptation loop

If a given element requires modification of the degree of polynomials in order to improve the quality of approximation, one of the three described scenarios is possible, increasing the degree of base functions in the direction of the axis x, increasing the degree of basic functions in the direction of the axis y, or raise the step in both directions.
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 (counted over all elements). This is, of course, an arbitrarily accepted value. In the proposed version of the algorithm, we first try to adapt in both directions (raise the degree of the polynomial by 1 in both directions), and then (if it is not indicated) we try to raise the degree of the polynomial in one of the two directions. This is, of course, an example version of the algorithm, various modifications are possible. The most advanced version of the adaptive algorithm is the hp adaptive algorithm described in another module.
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.
When an element patch consists of one element, the base functions spread over this element correspond to the classical p-adaptation algorithm working on Lagrange polynomials. This is because the node vectors in which each element is separated

C^0 separate, as is the case with classical Lagrange polynomials. Both bases then generate an identical space of polynomials. The mathematical properties of the p-adaptation algorithm were first described by Prof. Ivo Babuśka of the University of Texas at Austin
[1]

Bibliografia

1. Ivo Babuśka, B. Szabo, I. N. Katz: The p-version of the ?nite element method, SIAM Journal of Numerical Analysis, Society for Applied and Industrial Mathematics 1981, dostęp:18.10.2019

Utworzona przez admin. Ostatnia aktualizacja: Środa 28 z Październik, 2020 08:37:37 UTC przez paszynsk@agh.edu.pl. Autor: Maciej Paszynski