Pavel Kůs

Integration in higher-order finite element method in 3D

In: Jan Chleboun and Petr Přikryl and Karel Segeth and Jakub Šístek (eds.): Programs and Algorithms of Numerical Mathematics, Proceedings of Seminar. Dolní Maxov, June 6-11, 2010. Institute of Mathematics AS CR, Prague, 2010. pp. 131–136.

Persistent URL: http://dml.cz/dmlcz/702751

INTEGRATION IN HIGHER-ORDER FINITE ELEMENT METHOD IN 3D*

Pavel Kůs

1 Introduction

Integration of higher-order basis functions is an important issue, that is not as straightforward as it may seem. In traditional low-order FEM codes, the bulk of computational time is a solution of resulting system of linear equations. In the case of higher-order elements the situation is different. Especially in three dimensions the time of integration may represent significant part of the computation.

In first part of the text we describe Gauss quadrature and product quadrature rules on the reference brick. In Section 4.1 we describe algorithm calculating the local stiffness matrix en bloc, which allows to save a lot of calculations that would be repeated for many of the integrals of the stiffness matrix. Calculation is than much faster thanks to creation of auxiliary fields and multiple use of the values, which is shown in Section 5.

2 Gauss quadrature rules

The choice of quadrature type is very important. Even though two quadrature rules integrate exactly polynomials up to certain order, their performance can differ significantly when integrating non-polynomial functions (which is in reality always the case, since the inverse Jacobi matrix is non-polynomial for general mesh elements). The usual choice for higher-order integration are Gauss quadrature rules. A 1D integral over the segment $(-1, 1)$ is then approximated by the formula

$$\int_{-1}^{1} f(\xi)\mathrm{d}\xi \approx \sum_{i=1}^{n} w_{n,i} f(\xi_{n,i}), \tag{1}$$

where $\xi_{n,i}$ and $w_{n,i}$ are integration points and weights.

3 Product quadrature rules

Since the integration is performed on reference element, which is a cube in our case, the most natural choice of the integration rules is to use tensor products of 1D Gauss rules described in the previous section. A construction of such integration rules is described in [4].

*This work was supported by grant GAAVCR IAA100760702.

3.1 Computational cost of the integration

Let us estimate the computational cost of calculation of the local stiffness matrix. Consider hexahedral element. In *hp*-FEM it is usually equipped by basis functions constructed as products of 1D polynomials of degrees up to p. In total there are $(p+1)^3$ basis functions. In order to evaluate the local stiffness matrix, we have to calculate integral from the weak form for each pair of basis functions. Therefore we have $(p+1)^3 \times (p+1)^3$ integral evaluations. Integrand is always a product of two basis functions, its polynomial order therefore is up to $2p$ in each direction. Quadrature rule that will calculate integrals exactly has approximately p^3 points (each 1D rule has approximately p points). Thus, calculation of one integral costs $O(p^3)$ function evaluations. Since we have to do $(p+1)^3 \times (p+1)^3$ such calculations, total asymptotic complexity of the evaluation of the local stiffness matrix is $O(p^9)$.

It is obvious, that this is extremely unfavorable and makes assembling procedure very time-consuming. For the numerical solution of partial differential equations in more than 3 dimensions, this estimate is even more severe and makes it virtually impossible to use such integration. For truly high-dimensional calculations, which are becoming more and more desirable for example for financial problems, completely different ways towards estimation of the values of the integrals, such as Smoljak's schemes are used. For the main idea see Section 4.2.

3.2 Hierarchical elements

In the following we describe several ideas how to make the calculation more economical. If we use hierarchical rather then nodal basis, the basis of an element of order p is obtained by adding several polynomial functions of order p to the basis of an element of order $p-1$. Therefore, the basis consist of polynomials of various orders from 1 up to p and obviously it would be waste to integrate product of two low-degree polynomials with quadrature rule which is exact for product of polynomials of degree p. We consider basis functions in the form (2).

Assume we have to calculate product of two functions of degrees (p_x, p_y, p_z) and (q_x, q_y, q_z). Obviously, the rule capable of exact calculation is of order $(p_x + q_x, p_y + q_y, p_z + q_z)$. However, using such rules has slight drawback. When we calculate the value of the integral, we have to store precalculated values of all shape functions in all integration points of the particular rule. If we had precalculated values of all shape functions for all rules of order (p_x, p_y, p_z), $p_x, p_y, p_z \in \{1 \dots P\}$, where P is the maximal degree of polynomials used in the basis, the size of the tables would occupy a big portion of the computer memory. Possible solution of this problem is to use only quadrature rules with the same order in all directions, i.e. instead of a rule of order (p_x, p_y, p_z) we use a rule of order (p_m, p_m, p_m), where $p_m = \max(p_x, p_y, p_z)$ even though it has more points than necessary.

4 Alternative approaches to quadrature

In the previous section we described how a simple numerical quadrature works. We have seen, however, that this approach may lead to quadrature rules with very high number of integration points. In this section we want to describe two different approaches. The first is based upon the works [2], [1]. Ideas used there for 2D are adapted to 3D case and to different technique of construction of basis functions, which allows it's substantial simplification.

The second alternative is presented mainly for reference. Smoljak's schemes are used for integration in partial differential equations in more dimensions, where all conventional approaches fail due to the "curse of dimensionality".

4.1 Reordering of quadrature

We will use the fact, that both basis functions and integration rules are constructed as cartesian products of 1D functions and integration rules. Thanks to this structure, we can reorder the whole calculation, save some results into auxiliary fields and use them for more integrals of the stiffness matrix.

4.1.1 General algorithm

In the articles [2], [1], the authors distinguish between vertex, edge and bubble basis functions and use slightly different algorithm for each group. Our algorithm does not do that.

We consider basis functions on the reference domain $K = [-1, 1]^3$ in the form

$$F_{k_1,k_2,k_3}(\xi^1, \xi^2, \xi^3) = f^1_{k_1}(\xi^1) f^2_{k_2}(\xi^2) f^3_{k_3}(\xi^3), \tag{2}$$

where $(k_1, k_2, k_3) \in M = \{1, \ldots, n_1\} \times \{1, \ldots, n_2\} \times \{1, \ldots, n_3\}$. Our goal is to calculate all integrals

$$\int_K F_k(\xi) F_{k'}(\xi) Z(\xi) \, \mathrm{d}\xi, \tag{3}$$

where $k, k' \in M$. $Z(\xi)$ stands for the rest of the integrand independent on the basis functions. It can be Jacobian of reference mapping, material parameter or anything else. Of course this part of the integrals does not have product structure like the basis functions, but, on the other hand, is the same for all integrals calculated. The integrals will be approximated by one quadrature rule obtained as a product of three 1D rules with sufficiently high order in each direction. Individual 1D rules may have different order:

$$\begin{aligned} R_1 &= \{(w^1_i, \xi^1_i), i = 1, \ldots, m_1\}, \\ R_2 &= \{(w^2_i, \xi^2_i), i = 1, \ldots, m_2\}, \\ R_3 &= \{(w^3_i, \xi^3_i), i = 1, \ldots, m_3\}, \end{aligned}$$

where w^j_i stands for weight and ξ^j_i for integration point. The compound rule has then the form:

$$R = \{(w^1_{i_1} w^2_{i_2} w^3_{i_3}, \ (\xi^1_{i_1}, \xi^2_{i_2}, \xi^3_{i_3})), \ \ i_1 = 1, \ldots, m_1, \ i_2 = 1, \ldots, m_2, \ i_3 = 1, \ldots, m_3\},$$

the number of integration points being $m = m_1 m_2 m_3$. The integral from (3) can be approximated as

$$\sum_{i=1}^{m} w_i F_k(\xi_i) F_{k'}(\xi_i) Z(\xi_i). \tag{4}$$

Using the product structure of basis functions and integration rules, the latest can be expanded to

$$\sum_{i_1=1}^{m_1} \sum_{i_2=1}^{m_2} \sum_{i_3=1}^{m_3} w_{i_1}^1 w_{i_2}^2 w_{i_3}^3 \; f_{k_1}^1(\xi_{i_1}^1) f_{k_2}^2(\xi_{i_2}^2) f_{k_3}^3(\xi_{i_3}^3) \; f_{k_1'}^1(\xi_{i_1}^1) f_{k_2'}^2(\xi_{i_2}^2) f_{k_3'}^3(\xi_{i_3}^3) Z(\xi_{i_1}^1, \xi_{i_2}^2, \xi_{i_3}^3). \tag{5}$$

Now the summation can be reordered:

$$\sum_{i_1=1}^{m_1} w_{i_1}^1 f_{k_1}^1(\xi_{i_1}^1) f_{k_1'}^1(\xi_{i_1}^1) \sum_{i_2=1}^{m_2} w_{i_2}^2 f_{k_2}^2(\xi_{i_2}^2) f_{k_2'}^2(\xi_{i_2}^2) \sum_{i_3=1}^{m_3} w_{i_3}^3 f_{k_3}^3(\xi_{i_3}^3) f_{k_3'}^3(\xi_{i_3}^3) Z(\xi_{i_1}^1, \xi_{i_2}^2, \xi_{i_3}^3). \tag{6}$$

Let us introduce auxiliary field $G(k_3, k_3', i_1, i_2)$, where

$$G(k_3, k_3', i_1, i_2) = \sum_{i_3=1}^{m_3} w_{i_3}^3 f_{k_3}^3(\xi_{i_3}^3) f_{k_3'}^3(\xi_{i_3}^3) Z(\xi_{i_1}^1, \xi_{i_2}^2, \xi_{i_3}^3). \tag{7}$$

It is important to realize, that the just defined term really depends only on k_3, k_3', i_1 and i_2. Indeed, all terms depending on k_1, k_1', k_2 and k_2' were put in front of the last sum and i_3 is being summed over.

Similarly, let us introduce another auxiliary field $H(k_2, k_2', k_3, k_3', i_1)$:

$$H(k_2, k_2', k_3, k_3', i_1) = \sum_{i_2=1}^{m_2} w_{i_2}^2 f_{k_2}^2(\xi_{i_2}^2) f_{k_2'}^2(\xi_{i_2}^2) G(k_3, k_3', i_1, i_2). \tag{8}$$

This field depends also on k_2 and k_2', but, thanks to the summation, does not depend on i_2. Now the integral (3) can be approximated as

$$\int_K F_k(\xi) F_{k'}(\xi) Z(\xi) \, \mathrm{d}\xi \approx \sum_{i_1=1}^{m_1} w_{i_1}^1 f_{k_1}^1(\xi_{i_1}^1) f_{k_1'}^1(\xi_{i_1}^1) H(k_2, k_2', k_3, k_3', i_1) \tag{9}$$

When generating the matrix of the integrals, we first precalculate the field G, than the field H and finally use it to calculate all the integrals (9), where $k, k' \in M$.

4.1.2 Asymptotic analysis

Now let us estimate the amount of work needed to generate the stiffness matrix. The numerical comparisons are presented in Section 5, here we want to do just a rough estimate. As in Section 3.1, we assume, that the polynomial degree of our basis functions is up to p in each direction. Therefore we have p^3 functions and the 1D integration rules, that comprise the final integration rule, have approximately p integration points.

In Section 3.1, we approximated the work needed to generate the stiffness matrix to $O(p^9)$. In the algorithm described above, we first precalculate field G, which requires $O(p^4)$ work. Then the field H is precalculated, which requires $O(p^5)$. That should be negligible in comparison with the main part, which are calculations using the formula (9). There are p^3 functions, therefore we have to calculate p^6 integrals. But in the formula (9) there is only one summation, with respect to i_1. Other summations are hidden in the auxiliary fields. Therefore the complexity of this part is $O(p^7)$. Comparisons of real number of operations needed to calculate the matrix will be presented in Section 5.

4.2 Sparse schemes

The idea of sparse schemes was first introduced by Smolyak in [3]. The goal of this approach is to construct an integration grid, similar to the simple product grid, but with fewer points. The reason, why this is possible, is that slight under-integration does not always spoil the convergence.

From the experiments and comparisons we made it seems that this approach is not the most successful for problems in three dimensions. It's role starts to be vital for problems in much more dimensions, which arise in various fields including financial math. Sparse grids seems to be the only method capable to cope with the "curse of dimensionality", when number of integration points rise exponentially with number of dimensions.

5 Comparisons

In this section we want to compare different approaches to quadrature with respect to number of operations needed.

5.1 CPU time of assembling

The solving process in our code has two main parts, assembling and solving the stiffness matrix. A CPU time needed to perform each part very strongly depends on the problem setting. It depends not only on the number of elements of the mesh and the polynomial order used in the finite element space, but also on the structure of the mesh and use of hanging nodes. Often the assembling time exceeds the time needed to solve the resulting linear system, so faster quadrature can be very welcomed in some cases.

5.2 Performance of different quadrature techniques

In Figure 1 we can see a comparison of number of operations needed to assembly a mass matrix of the element of various orders. On the graph we can see ratios of number of operations of individual methods with respect to the simple product method (it is therefore 1 for all polynomial orders.)

The usefulness of the faster quadrature depends on the order used, but even for order 5 we get ten times faster algorithm, comparing to the integration of each integral with optimal, but isotropic order.

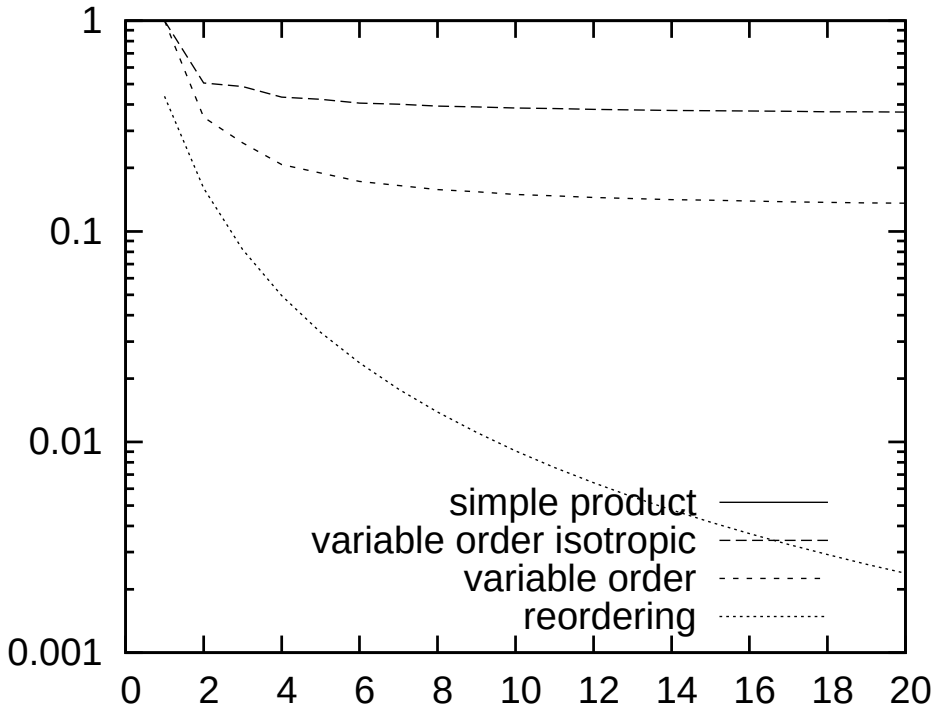

Fig. 1: *Comparison of performance of described methods with respect to the simple product method. On the x axis is order of an element, on the y axis quotient of number of operations of each method.*

5.3 Conclusions

We have shown, that the concept of reordering of summation works well and decreases number of operations needed to construct the local stiffness matrix. It's effect grows with growing order of an element. Even though incorporating into the code might bring certain complications, it is definitively worth considering.

References

[1] Eibner, T. and Melenk, J.M.: Fast algorithms for setting up the stiffness matrix in *hp*-FEM: a comparison. Numerical Analysis Report 3/05 .

[2] Melenk, J.M., Gerdes, K., and Schwab, C.: Fully discrete *hp*-finite elements: Fast quadrature. Research Report No. 99-15 (1999).

[3] Smolyak, S.A.: Quadrature and interpolation formulas for tensor product of certain classes of functions. Dokl. Akad. Nauk (1963), 240–243.

[4] Solin, P., Segeth, K., and Dolezel, I.: *Higher-Order Finite Element Methods.* Chapman & Hall/CRC Press, Boca Raton, 2004.