STUDY ON FINITE DEFORMATION FINITE ELEMENT ANALYSIS ALGORITHM OF TURBINE BLADE BASED ON CPU+GPU HETEROGENEOUS PARALLEL COMPUTATION

by

Tian-Yuan LIU, Hao DANG, and Yong-Hui XIE*

School of Energy and Power Engineering, Xi'an Jiaotong University,
Xi'an, China

Original scientific paper
DOI: 10.2298/TSCI16S3823L

Blade is one of the core components of turbine machinery. The reliability of blade is directly related to the normal operation of plant unit. However, with the increase of blade length and flow rate, non-linear effects such as finite deformation must be considered in strength computation to guarantee enough accuracy. Parallel computation is adopted to improve the efficiency of classical non-linear finite element method and shorten the blade design period. So it is of extraordinary importance for engineering practice. In this paper, the dynamic partial differential equations and the finite element method forms for turbine blades under centrifugal load and flow load are given firstly. Then, according to the characteristics of turbine blade model, the classical method is optimized based on central processing unit + graphics processing unit heterogeneous parallel computation. Finally, the numerical experiment validations are performed. The computation speed of the algorithm proposed in this paper is compared with the speed of ANSYS. For the rectangle plate model with mesh number of 10 k to 4000 k, a maximum speed-up of 4.31 can be obtained. For the real blade-rim model with mesh number of 500 k, the speed-up of 4.54 times can be obtained.

Key words: *central processing unit, graphic processing unit, heterogeneous parallel computation, finite deformation, finite element analysis algorithm*

Introduction

Blade is one of the core components of turbine machinery, which sustains a variety of loads and works at poor working conditions. Nevertheless, with the increase of blade length and flow rate, non-linear effects such as large deformation must be considered in strength design computation to guarantee the accuracy, which will significantly increase the calculation time of design process. However, the use of parallel computing can greatly improve the efficiency of conventional finite element method (FEM) and shorten the design period of blades. Hence, it is of extraordinary importance on engineering practice.

The most time-consuming part of conventional central processing unit (CPU) parallel FEM is generally the computation and assembly of element matrix and iteration solving of non-linear equations. For the heterogeneous parallel acceleration algorithm of the former, Zhisong *et al.* [1] presented a new data mapping method. Compared to conventional CPU serial finite element (FE) program, this new method acquired an 81-fold speed-up when matrix assembly.

* Corresponding author; e-mail: yhxie@mail.xjtu.edu.cn

Karatarakis *et al.* [2] put forward a stiffness matrix integral computation and assembly method for element free Galerkin meshless methods and obtained a 202-fold speed-up of a turbine blade, in which the parallel strategy of matrix blocking and dense matrix can be utilized in FEM matrix computing and assembly. As for turbine blades, the stiffness matrix of the structure is always too ill-conditioned, which makes the linear equations solving in each iteration step extraordinary slower than that based on direct method of matrix decomposition-decimation. The ineffective sparse lower-upper (LU) decomposition method must be adopted in graphics processing unit (GPU) acceleration [3, 4], in which the parallel optimization usually implements substructure method. Actually, the study in non-linear dynamics of a mistuned blade based on substructure method in [5] is more effective when disposing strength vibration problem of a blade. The optimization of the fan blade based on substructure method in [6] is also practical. Adopting this method as the foundation to execute heterogeneous parallel optimization of the traditional algorithm can improve computing speed and shorten design period.

On the basis of the discussion, firstly in this paper, the general kinetic equation for the turbine blade under actual load is established. Afterwards, considering the characteristic of turbine blade static strength FE computation, the heterogeneous parallel optimization is conducted based on CPU+GPU. A new algorithm is designed to increase the computing and assembly speed of stiffness matrix, which combines substructure method and Newton method for acceleration of non-linear iteration computation. Finally, the numeric simulation test is implemented.

Mathematical model

The dynamic partial differential equations for turbine blades under centrifugal load and flow load are given by [7, 8]:

$$\begin{cases} \rho\ddot{\vec{u}} = \nabla(\mathbf{D}\boldsymbol{\varepsilon}) + f, & X \in \Omega \\ \vec{u} = C, & X \in \Gamma_1 \\ f = f_{\vec{b}} + f_c, & X \in \Gamma_2 \\ f = f_c, & X \in \Omega - \Gamma_2 \end{cases} \tag{1}$$

where $\vec{u}$ is the displacement vector, Ω – the computational domain of the blade, X – the point in this domain, Γ_1 – the fixed area of the blade root or rim, Γ_2 – the part of the blade body flushed by the working fluid, **D** – the fourth-order constitutive stiffness tensor, the density of the blade material. The centrifugal load can be expressed as $f_c = \rho\omega^2\vec{r}$, where $\vec{r}$ is the projection of the X position vector, ω – the angular velocity of the blade, and the flow flushing load is $f_{\vec{b}} = p\vec{n}$, where p is the pressure of flushing face, and $\vec{n}$ – the normal vector of flushing face.

In a real process, long blades sustain large load, which usually produces large deformation. The strain tensor in eq. (1) should be expressed as Green-strain according to Lagrange description.

$$\boldsymbol{\varepsilon} = \frac{1}{2}[\nabla\vec{u} + (\nabla\vec{u})^{\mathrm{T}} + \nabla\vec{u}(\nabla\vec{u})^{\mathrm{T}}] \tag{2}$$

Consider the formulation of shape function $X = [N]\{u\}^{\mathrm{e}}$, then its finite element analysis (FEA) form is:

$$\sum_{\Omega}[\mathrm{M}]^{\mathrm{e}}\{\ddot{\vec{u}}\}^{\mathrm{e}} = \sum_{\Omega}[\mathrm{K}]^{\mathrm{e}}\{\vec{u}\}^{\mathrm{e}} + \sum_{\Omega}(\{\vec{F}_c\}^{\mathrm{e}} + \{\vec{F}_q\}^{\mathrm{e}}) \tag{3}$$

where $[M]^e$ is the mass matrix for the element, $[K]^e$ the stiffness matrix, $\{\vec{F}_c\}^e$ the vector of the centrifugal force, and $\{\vec{F}_b\}^e$ is the vector of the flow induced force for the element. In the static analysis of the turbine blade, the left of the eq. (3) is a zero vector. As a result of $[K]^e$ is the function of variables $\vec{u}$, the original equation has become a non-linear partial differential equation, which the Newton-Raphson method is usually used to solve [9, 10].

Algorithm optimization

The non-linear FEA of finite deformation for turbine blade can be treated as the problems of compute-intensive and memory-constraint. The major time-consuming steps are the calculations of element matrix computation-assembly and the solving of non-linear equations. The former one takes 20%~25% of calculation time and the solving of non-linear equations takes 60%~70%. The optimization of these two steps is illustrated in detail.

Parallel strategy of element matrix computation and assembly

Element computation includes the calculation of element matrix and vector, which is of natural parallelism. While common nodes exist among each element, the overall matrix of these nodes needs superposition from the node on element matrix, which causes thread interference and is unfavorable for the computational speed. In order to distinguish the elements with common nodes, every element is dyed and two elements with shared node appear in different colors, which divide all elements into several sets. The two arbitrary elements in same color set share no common nodes.

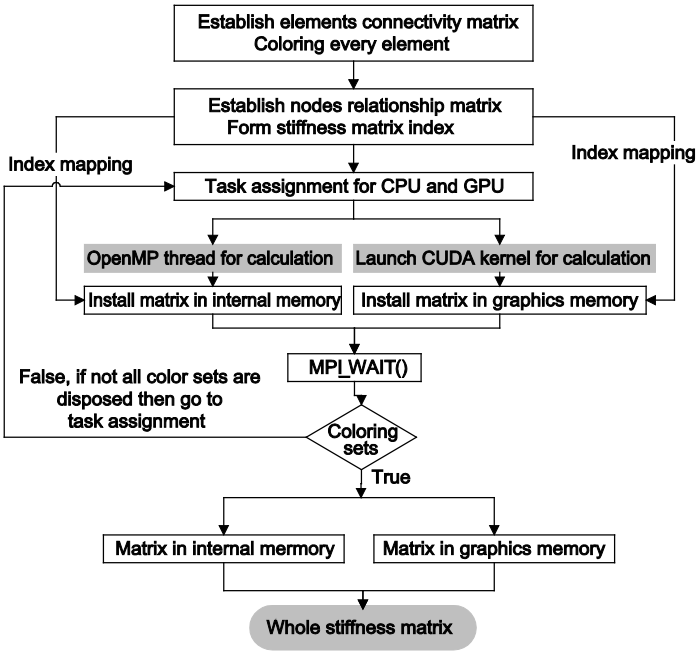

Figure 1. Flow chart of matrix computation and assembly algorithm

The essence of element matrix computation is multiplication of small-scale matrix. However, these matrices are not completely dense but of certain sparse and symmetry [11]. We can further arrange computation on the basis of specialty to decrease computational complexity and raise parallel efficiency. The multiplication of each intensive micro block matrix

can conduct the fine-grained parallel computation in CUDA at atomic level. It can also be directly calculated on CPU by invoking the Blas library, then utilize OpenMp to run multi-core parallel [12]. The algorithm is shown fig. 1.

The solving method of non-linear equations

Considering the mesh features in a general blade strength analysis, the mesh is dense in the region of stress concentration (*i. e.* blade root), while its deformation is generally small. The deformation at blade body is large while the strain/stress is small. The number of mesh or nodes at blade root usually accounts for 60%~80% of the whole structure. The specific solving procedure is shown in continuation of the paper. There is a model of N blade-rim in fig. 2. With substructure method, it has been divided into N + 1 domains. The first N regions are consist of the mesh of blade bodies and the N + 1 is where root-rim combine. The dyed regions as the fig. 2 on the right are the boundary of substructures.

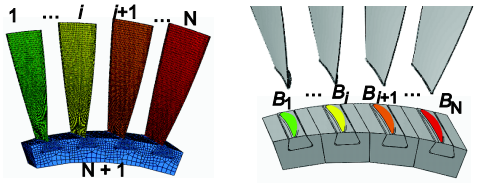

Figure 2. Substructure regions of turbine blades

Without loss of generality, according to substructure method, the tangent stiffness matrix equation takes the form when N is assumed as 2, namely:

$$[K_T]\{d\vec{u}\} = \begin{bmatrix} K_1 & & & T_{B1} & \\ & K_2 & & & T_{B2} \\ & & K_R & T_{R1} & T_{R2} \\ T_{B1}^{T} & & T_{R1}^{T} & B_1 & \\ & T_{B2}^{T} & T_{R2}^{T} & & B_2 \end{bmatrix} \begin{Bmatrix} d\vec{u}_1 \\ d\vec{u}_2 \\ d\vec{u}_R \\ d\vec{u}_{\bar{b}1} \\ d\vec{u}_{\bar{b}2} \end{Bmatrix} = \begin{Bmatrix} d\vec{F}_1 \\ d\vec{F}_2 \\ d\vec{F}_R \\ d\vec{F}_{\bar{b}1} \\ d\vec{F}_{\bar{b}2} \end{Bmatrix} \tag{4}$$

where K_R on the principal diagonal is the stiffness sub-matrix of blade region *i*, K_R – the stiffness sub-matrix at blade root-rim, B_i – the stiffness sub-matrix at boundaries, and the rest matrixes T are the incidence sub-matrixes between substructures. According to sub-matrix decomposition algorithm [6, 13], then we have:

$$K_R d\vec{u}_R = \vec{F}_R - T_{R1}(K_{B1}^{-1})(\vec{F}_{\bar{b}1} - T_{B1}K_1^{-1}\vec{F}_1 - T_{R1}^{T}d\vec{u}_R) -$$
$$-T_{R2}(K_{B2}^{-1})(\vec{F}_{\bar{b}2} - T_{B2}K_2^{-1}\vec{F}_2 - T_{R2}^{T}d\vec{u}_R)$$

where $K_{B1} = B_1 - T_{B1}^{T}K_1^{-1}T_{B1}, K_{B2} = B_2 - T_{B2}^{T}K_2^{-1}T_{B2}$.

Considering the displacement modification $d\vec{u}_R$ at blade root is generally small, in the Newton iteration step, it can be ignored. Then, it is written as:

$$d\vec{u}_R \approx K_R^{-1}[\vec{F}_R - T_{R1}K_{B1}^{-1}(\vec{F}_{\bar{b}1} - T_{B1}K_1^{-1}\vec{F}_1) - T_{R2}K_{B2}^{-1}(\vec{F}_{\bar{b}2} - T_{B2}K_2^{-1}\vec{F}_2)] \tag{5}$$

The displacement vector modification:

$$\begin{cases} d\vec{u}_{\bar{b}1} = K_{B1}^{-1}(\vec{F}_{\bar{b}1} - T_{B1}K_1^{-1}\vec{F}_1 - T_{R1}^{T}d\vec{u}_R) \\ d\vec{u}_{\bar{b}2} = K_{B2}^{-1}(\vec{F}_{\bar{b}2} - T_{B2}K_2^{-1}\vec{F}_2 - T_{R2}^{T}d\vec{u}_R) \\ d\vec{u}_1 = \vec{F}_1 - T_{B1}d\vec{u}_{\bar{b}1}u_{s-1} \\ d\vec{u}_2 = \vec{F}_2 - T_{B2}d\vec{u}_{\bar{b}2} \end{cases} \tag{6}$$

The decomposition of K_R is only conducted at the beginning of the iteration, so it is stored. Therefore, we only need to solve the equation consist of K_1, K_2, K_{B1}, K_{B2} in each iteration step in practice. The modified Newton method can be achieved with the following algorithm:

(1) According to $\vec{u} = 0$, compute the initial tangent stiffness matrix K_{T0} and load vector $\vec{F}_0$, which are constituted as eq. (4).
(2) Acquire the initial stiffness matrix K_R of the blade root-rim, conduct LU decomposition and store in internal memory.
(3) The matrix decomposition of K_{i0} and K_{Bi0}, and obtain the linear displacement $\vec{u}_0$.
(4) Access Newton iteration step s (s = 1, 2, 3,…).
(5) Calculate the new tangent stiffness matrix K_{TS} and secant stiffness matrix K_{SS} in accordance with $\vec{u}_{s-1}$.
(6) Compute force residual vector $\vec{R}_F = K_{GS}\vec{u}_{s-1} - \vec{F}_0$. If $\|\vec{R}_F\| < err$, then the iteration and the solution vector is $\vec{u}_{s-1}$, else continue step 7.
(7) Solve the matrix decomposition of K_{is} and K_{Bis} in parallel.
(8) With the usage of decomposition of K_R, K_{is}, and K_{Bis}, compute the displacement modification $d\vec{u}_{s-1}$ based on eqs. (5) and (6).
(9) Acquire new displacement $\vec{u}_{s-1} = \vec{u}_s + d\vec{u}_{s-1}$, $s = s + 1$, return to step 4.

Results and discussion

Based on the previous algorithm, a FEA program (hereinafter referred to as TbFEA program) which supports CPU+GPU heterogeneous parallel computing is established in this paper and the compiling environment is Windows 7, Visual studio2013, CUDA7.5, and MPICH2. The computing performance in this paper is all tested upon a single computational node (one Tesla k20c/dual processors Intel Xeon E2650/128G physical internal memory).

Geometrical non-linear analysis of a rectangular plate

Firstly, a simple rectangular plate is chosen to test the precision and acceleration effect of TbFEA. The size of the rectangular plate is shown in fig. 3. Material parameters are: elastic modulus 210 GPa, Poisson's ratio 0.3, and density 7850 kg/m^3. As for boundary conditions, the nodes on YOZ plane are fully constrained and the force of 17000 N in Z-direction is applied evenly at the end of the plate. In order to simulate the mesh characteristic of a real blade, the

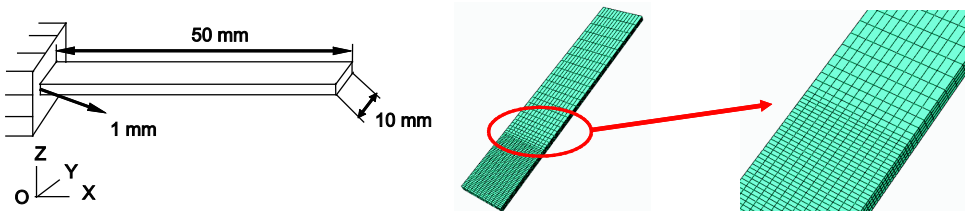

Figure 3. Rectangular plate geometry/mesh

mesh number of a quarter of length near the constraint wall is three quarters of the totality (small deformation and large stress in this region) while the other three quarters of length away from the wall account for a quarter; the total mesh number for calculation is set in tab. 1.

Table 1. Computation time for non-linear equation with different mesh number

Time [s]	10 k	50 k	100 k	200 k	500 k	1000 k	2000 k	3000 k	4000 k
Method 1	3.128	30.51	126.5	384.4	1374	4215	15521	54655	87785
Method 2	3.474	24.19	87.45	262.2	914	2814	10116	33647	52249
Method 3	2.196	16.06	56.72	158.1	553	1623	5918	22210	37409
Method 4	4.516	11.67	31.18	80.49	252	713	2336	8422	16175

The computational time of TbFEA program and ANSYS with different mesh number and the result is shown in tab. 1. Method 1 is performed with only 2 CPU in ANSYS and Method 2 is two CPU+GPU, these former two adopt full Newton method as algorithm; Method 3 is conducted in TbFEA with substructure method and full Newton method. Method 4 is in TbFEA with substructure method and modified Newton method for iteration. Figure 4 reflects the actual acceleration effect of the algorithm proposed in this paper, where S1 shows the ratio of Method 1 and Method 2 in computational time and S2 represents that of Method 1 and Method 3. In order to impartially compare the acceleration effect of TbFEA when using the same hardware, ST represents the ratio of Method 2 and Method 4 in computational time. The computation time of four methods with 2000 k is compared in right of fig. 4.

It is indicated from fig. 4 that the speed-up of ANSYS with GPU acceleration, compared with the mere application of CPU, preserves around 1.5 when the mesh number is approximately 1000 k. With further improvement of mesh number, the speed-up slightly increases. However, TbFEA with substructure method can preferably accelerate the computation. With mesh number of 2000 k, that the speed-up is about 2.5 and when above 2000 k, it descends, since the decomposition of the global stiffness matrix has exceeded internal memory and that of substructure has also exceeded graphical memory, the problem now transforms to memory constraint; when applying modified Newton method, compared with ANSYS with GPU, the maximum speed-up can be 4.31. As a matter of fact, the speed-up has reached up to 6.64 in comparison to ANSYS with only two CPU. When the mesh number surpasses 2000 k, the speed-up as well declines. Nevertheless, the speed-up can further increase with sufficient internal memory.

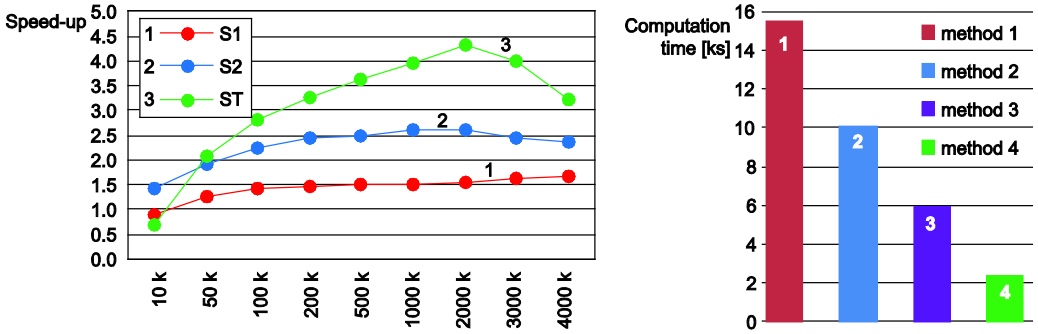

Figure 4. The acceleration effect of GPU+CPU

Finite deformation analysis of a real blade model

A real free blade-rim model of three was analyzed in fig. 5. The blade root is fir-tree of four teeth. The blade height is 1.8 m. Material parameters of the blade are elastic modulus 210 GPa, Poisson's ratio 0.3, and density 7850 kg/m^3. The working condition is centrifugal load of 1500 rpm. The flow exciting force on the blade surface is 2 MPa. This paper does not involve contact analysis. For the analysis of contact surface, a simple way is applied, *i. e.*, coupling all degrees of freedom of nodes on blade root-rim. The mesh generation of the blade adopts structured grid. A thin layer of tetrahedral mesh is employed only in the transition region between the blade body and the blade root. The blade root-rim shows apparent stress concentration and the mesh number is approximately 400 k. While the blade body has large deformation and low stress, the mesh number is about 100 k.

Figure 6 shows the positions from the middle blade, which are used to compare the stress result of ANSYS and TbFEA. Von-Mises stress distributes along the profile lines are shown in figs. 7 and 8. Both the two parts produce stress concentration and need special attention. The stress curves of the examine positions from TbFEA and ANSYS basically coincide, which illustrate that the precision of TbFEA satisfies the computing requirement. Figure 9 shows overall stress distribution of the blade root. In this blade model, the maximum stress of blade root is 403.01 MPa at the fourth bearing teeth fillet of suction surface, as for the fillet on blade root platform, is 293.13 MPa of pressure surface.

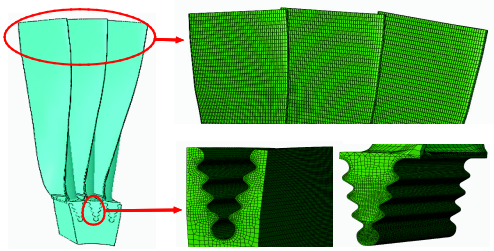

Figure 5. Turbine blade mesh generation

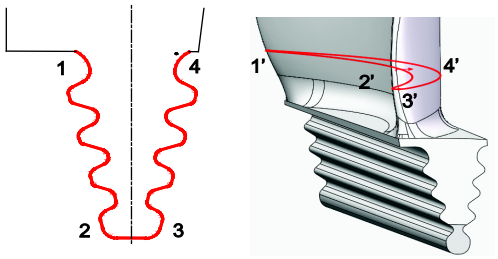

Figure 6. Stress examine positions

The ANSYS with full Newton method needs nine steps in total. It takes 1335 seconds to solve with two CPU, and 1132 seconds with two CPU+GPU; while TbFEA with modified Newton method proposed in this paper needs 21 iteration steps and takes 249 seconds when parallel computing with two CPU+GPU at the same time. The speed-up is 4.54, which demonstrates apparent effect of acceleration.

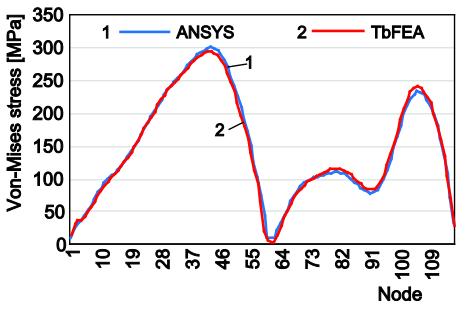

Figure 7. Stress of the blade body profile line

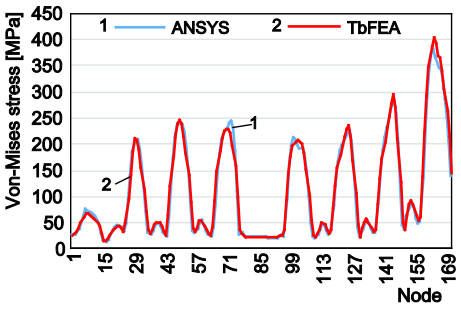

Figure 8. Stress of the root profile line

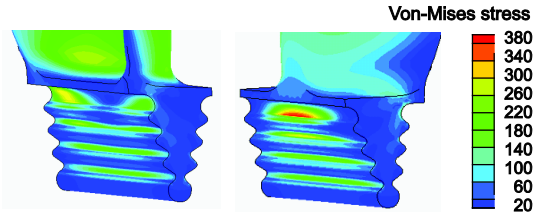

Figure 9. Von-Mises stress distribution of the blade root

Conclusion

In this paper, an optimization algorithm for classical FEM of turbine blade is proposed based on CPU+GPU heterogeneous parallel computation. The parallelism efficiency of element matrix computation and assembly is improved by element dyeing along with utilization of the element matrix sparsity and symmetry. A new nonlinear method that combines substructure method and newton method is provided. This method can decrease the solving time of nonlinear equations. Finally, the numerical experimental verification has been performed, and the computation speed of the algorithm proposed in this paper is compared with ANSYS. For the rectangle plate model with mesh number of 10 k to 4000 k, a maximum speed-up of 4.31 can be obtained. For the real blade-rim model with mesh number of 500 k, the speed-up of 4.54 can be obtained.

Nomenclature

$\vec{b}$ – normal vector of flushing face, [–]
C – displacement constant, [m]
$\mathbf{D}$ – fourth-order stiffness tensor, [Pa]
$f_{\vec{b}}$ – flow flushing load, [Nm^{-3}]
f_c – centrifugal load, [Nm^{-3}]
p – pressure of flushing face, [Pa]
$\vec{r}$ – position vector, [m]
$\vec{u}$ – displacement vector, [m]
$\ddot{\vec{u}}$ – acceleration vector, [ms^{-2}]

Greel symbols

$\boldsymbol{\varepsilon}$ – strain tensor, [–]
ρ – material density, [kgm^{-3}]
Ω – computational domain, [–]
ω – angular velocity, [s^{-1}]

References

[1] Zhisong, F., *et al.*, Architecting the Finite Element Method Pipeline for the GPU, *Journal of Computational and Applied Mathematics*, *257* (2014), Feb., pp. 195-211
[2] Karatarakis, A., *et al.*, GPU-Acceleration of Stiffness Matrix Calculation and Efficient Initialization of EFG Meshless Methods, *Computer Methods in Applied Mechanics and Engineering*, *199* (2010), May, pp. 3305-3314
[3] Ren, L., *et al.*, Sparse LU Factorization for Parallel Circuit Simulation on GPU, IEEE, *Proceedings,* Design Automation Conference, San Francisco, Cal., USA, 2012
[4] Liu, L., *et al.*, A Highly Efficient GPU-CPU Hybrid Parallel Implementation of Sparse LU Factorization, *Chinese Journal of Electronics*, *20* (2012), 1, pp. 7-12
[5] Capiez-Lernout, C., *et al.*, Geometric Non-Linear Dynamic Analysis of Uncertain Structures with Cyclic Symmetry – Application to a Mistuned Industrial Bladed Disk, *Proceedings*, International Conference on Uncertainty in Structural Dynamics, Leuven, Belgium, 2014
[6] Chew, K., *et al.*, Structural Optimization and Parametric Study of Offshore wind Turbine Jacket Substructure, *Proceedings*, 23rd International Offshore and Polar Engineering Conference, Anchorage, Alas., USA, 2013, ISOPE-I-13-010
[7] Smith, I. M., *et al.*, *Programming the Finite Element Method*, 3rd ed., Beijing Electronic Industry, Beijing, 2005
[8] Xie, Y. H., *The Study for Fatigue Failure Life Estimation and Design Analysis System of Steam Turbine Blade,* Xi'an Jiaotong University, Xi'an, China, 1997
[9] Rene, B., *et al.*, *Non-Linear Finite Element Analysis of Solids and Structures*, 2nd ed., John Wiley & Sons, New York, USA, 2012
[10] Ted, B., *et al.*, *Non-Linear Finite Elements for Continua and Structures*, John Wiley & Sons Ltd., Chichester, UK, 2000

[11] Thomas, J. R., *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Dover, Mineola, N.Y., USA, 2000
[12] Thomas, R., *et al.*, *Parallel Programming For Multicore and Cluster Systems*, Springer, New York, USA, 2010
[13] Shen, L., Shusen Z., Study on Substructure Method for Torsional Vibration of a Branch Shafting System, *Journal of Vibration and Shock, 26* (2007), 10, pp. 148-151

Paper submitted: December 14, 2015
Paper revised: December 20, 2015
Paper accepted: December 25, 2015