JAM CM Journal of Applied Mathematics and Computational Mechanics 2017, 16(1), 109-116
www.amcm.pcz.pl p-ISSN 2299-9965
DOI: 10.17512/jamcm.2017.1.10 e-ISSN 2353-0588

REAL-TIME FINITE ELEMENT SIMULATIONS ON AN ARM MICROCONTROLLER

Waldemar Mucha

Institute of Computational Mechanics and Engineering, Silesian University of Technology
Gliwice, Poland
waldemar.mucha@polsl.pl

Received: 24 November 2016; accepted: 15 March 2017

Abstract. Real time processing enables online reactions to dynamic environment changes. The author focuses on possibilities of implementing the Finite Element Method (FEM) to real time algorithms on microcontrollers. The paper presents the current state of the Real Time Finite Element Method (RTFEM) and describes results obtained by the author. The RTFEM for microcontrollers (with examples of use cases) is presented and results of the computations for the chosen platform are given. The results consider optimization of RTFEM computational algorithms for a microcontroller taking into account their execution time. All the tests were performed on the ARM-CortexM4F based STM32F429ZIT6 microcontroller. The obtained results were compared, discussed and presented in the paper.

MSC 2010: *65N30*
Keywords: *finite element method, real time, embedded systems, microcontroller*

1. Introduction

The real time processing and analysis of signals, implemented on microcontrollers has been present in everyday life for years. The algorithms behind them are as simple as possible because of their low computational performance. The real time processing enables online reactions to dynamic environment changes. Author focuses on the possibilities of implementing the Finite Element Method (FEM) to real time algorithms on microcontrollers.

The following paper focuses on implementing Real Time Finite Element Method (RTFEM) computational algorithms on ARM microcontrollers by adjusting the algorithms to a specific structure and requirements of ARM architectures.

Real time computations and processing of data using microcontrollers is nowadays widely implemented in many areas of applications. Taking into account the architecture and low computational performance of microcontrollers, the algorithms behind them are as simple as possible. Real time algorithms enable immediate reactions to dynamic changes of the environment. The author of the paper

focuses on possibilities of implementing real time algorithms on microcontrollers with the Finite Element Method (FEM). Real time means a previously specified time interval in which the computations must be performed so the results would be available. The number of floating-point operations that need to be performed directly depends on the number of degrees of freedom of the model. The computational power of microcontrollers and embedded platforms is growing rapidly every year, opening up bigger and bigger possibilities for algorithms where a large number of floating point operations has to be performed.

In the literature, only a few examples can be found of applications of the Real Time Finite Element Method (RTFEM). The greatest problem in the RTFEM is the conflict between the accuracy of the results and the execution time of the algorithm. Applications found in the literature involve either RTFEM models where results are just very roughly estimated or metamodels based on previous FEM calculations. Examples of the applied RTFEM found in the literature consider surgery applications (simulations of cuts [1], plastic surgery [2], suturing [3] and other procedures), thermal simulations [4], recovering the force and location in an impact event [5], part inspection in manufacturing [6] and controlling elastic soft robots [7].

The purpose of implementing RTFEM algorithms to microcontrollers can be performing Hardware-in-the-loop (HIL) simulations. An example of applying RTFEM in HIL described in [8] is to use a materials testing machine controlled by a microcontroller. A physical part of the tested mechanical system is mounted in a dynamic materials testing machine, while the rest of the system is modelled using the FEM. In order to control the machine in a closed loop, the microcontroller has to perform dynamic FEM computations in real time. The algorithm is improved in [9] where mode superposition is used to speed up the real time computations. Paper [10] describes using an ARM microcontroller for optimization of industrial processes, where metamodels, evolutionary algorithms and artificial neural networks are used. Implementing the RTFEM to such microcontroller applications is just one step further, as the FEM is widely used in solving inverse problems like optimization or identification [11-13].

The following paper describes an algorithm for reducing computational time of FEM calculations conducted on ARM microcontrollers by replacing every element of floating-point data with integer data and introducing scaling. All the tests were performed on the ARM-CortexM4F based STM32F429ZIT6 microcontroller.

2. Code optimization of the RTFEM algorithm for the ARM microcontroller

In computer science, code optimization is the process of modifying the code to make the software work more efficiently or use fewer resources. The program can be optimized so that it executes faster, or operates with less memory storage or other resources.

Taking into account the low computational power of microcontrollers, the algorithms implemented on them should be optimized in respect of execution time related to the amount of floating point operations that need to be performed.

Application of the FEM is based on solving matrix equations. When considering mechanical analysis the equation is [14]:

$$\mathbf{Kq} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{M}\ddot{\mathbf{q}} = \mathbf{F}, \tag{1}$$

where **K** is the stiffness matrix, **C** is the damping matrix, **M** is the mass matrix, **q** is the vector of unknown displacements, $\dot{\mathbf{q}}$ is the velocity vector, $\ddot{\mathbf{q}}$ is the acceleration vector and **F** is the load vector.

Algorithms for solving equation (1) (for example the Central Difference Method, Newmark Method or Wilson Method) consist of solving this equation for every discrete moment of time, usually by reducing it to a simple matrix equation to be solved:

$$\mathbf{Ax} = \mathbf{b}, \tag{2}$$

where **x** is unknown.

When considering a static strength analysis of a structure using FEM, the equation (1) is from the beginning in the form (2):

$$\mathbf{Kq} = \mathbf{F}. \tag{3}$$

The first optimization procedure is the choice of a method to solve the FEM equation. The matrix equation (2) can be numerically solved by computing and applying the inverse matrix or by using other algorithms [15]. Since inversing matrices numerically is very computationally expensive and not stable, there is a choice between using direct and iterative methods [16]. To choose a good method for a specific job, one must consider the required speed and accuracy that are necessary. If great precision is required or the matrix is small (up to about a few hundred rows, depends of the matrix density) the direct methods are a much better choice than the iterative methods. For big matrices, iterative methods are usually faster but less precise than direct methods [17]. Iterative methods are especially suitable for sparse matrices [18], that usually occur in the FEM. The Gauss Elimination Method, Cholesky decomposition, LDLT decomposition are examples of direct methods. The Conjugate Gradient Method is, on the other hand, the most popular (in FEM applications) iterative method that applies to equations like (2), where the matrix **A** is symmetrical [19].

The second optimization procedure for FEM simulations on microcontrollers is the choice of data type format in which constants and variables in the algorithm are represented. Typically, microcontrollers have to offer two formats for floating-point variables and constants: single precision (*float*) and double precision (*double*). The author advises using the single precision format in typical applications where it is possible (the range of values is not exceeded and the precision is sufficient) because arithmetical operations on single precision values are faster. Another idea

to speed-up the algorithm is to represent every floating-point variable and constant with an integer data type format. It is possible if proper scaling is used.

As an example, the Newmark algorithm is taken into consideration [14]. The first step of representing all values in the integer format is to choose such a system of consistent units that will provide as much input and output data as possible with a positive order of magnitude. If not all input data are integer constants in the chosen consistent units system, they can be scaled.

When scaling input data, one must keep in mind that later when calculating stiffness and mass matrices from the scaled data, both matrices must be in the same scale. If they are, the calculated damping matrix (usually from stiffness matrix and mass matrix using the Rayleigh model) and effective stiffness matrix are also in the same scale. If the stiffness, mass and damping matrices are scaled by k_1 (in consequence the effective stiffness matrix is in scale k_1) and the load vector (in every time moment) is scaled by multiplying its elements by k_2, the initial conditions have to be scaled by quotient k_2/k_1. Only then the results of displacements, velocities and accelerations will be valid and also obtained in scale k_2/k_1. When introducing the abovementioned scales to the algorithm, one know that when performing a dividing operation on values represented by the integer format, the result of the operation is the quotient, and the remainder is lost, so the result is rounded always down. Taking that into consideration, k_2/k_1 proportion should be chosen carefully so the results would not be too inaccurate. The order of magnitude of the obtained scaled results from 3 to 8 usually works.

The final remark on adjusting the Newmark algorithm to integer operations is that the notation of equations for computing an effective stiffness matrix, effective load vector, acceleration vector and velocity vector should be changed to a form where there is only multiplication and division by integer values, as the integration parameters α and δ are usually smaller then 1. For example, if integration step dt is 10 ms, parameter α is 0.25 and parameter δ is 0.5, the formula to calculate every element of the effective stiffness matrix:

$$K_{eff_{ij}} = K_{ij} + \frac{1}{\alpha \cdot dt^2} M_{ij} + \frac{\delta}{\alpha \cdot dt} C_{ij} \tag{4}$$

should be changed to:

$$K_{eff_{ij}} = K_{ij} + \frac{M_{ij}}{25} + \frac{C_{ij}}{5} \tag{5}$$

with parameter k_1 just high enough not to lose too much accuracy while rounding down when performing integer divisions in equation (5).

3. Numerical examples

Two numerical examples of FEM computations were performed on ST-DISCOVERY evaluation board, equipped with the STM32F429ZIT6 microcontroller.

This device integrates ARM-CortexM4F based core (ARMv7M architecture), program and data memories, and all the necessary peripherals which make an autonomic computer system [20]. The algorithmswere implemented with use of C# .NET MF. The CPU core was clocked at 180 MHz, which is the maximal frequency for a ST32F429ZIT6 microcontroller.

Both FEM examples concern comparison between accuracy and computation time for different variables and constants data types representations. Both FEM examples consider a steel bridge (modeled by a steel truss: Young Modulus $E = 210{,}000$ MPa, cross section area $A = 2000$ mm^2, density $\rho = 0.0078$ g/mm^3). The selected system of consistent units in the FEM algorithms is: g, mm, ms, N, MPa.

A vehicle (with the mass of 80,000,000 g, represented by F_{veh}) is crossing the bridge (from point A to D) with the velocity of 15 mm/ms (Fig. 1). The load is distributed on two nodes (between which the vehicle is located, the forces are proportional to the distance from the nodes to the vehicle) or on one node if the vehicle's center of gravity is directly above it. The task is to find deformation of point B of the bridge.

Three variants of the algorithm are considered, where every floating point value is represented by: double-precision floating point format (*double*), single-precision floating point format (*float*) and integer format (*int*, with scaling applied). The obtained results from the algorithm were compared with results from commercial CAE program - Abaqus CAE 6.13-1 (implicit integration scheme was chosen, with identical time step as in the microcontroller - 20 ms).

Execution times for every variant of the algorithm are presented in Table 1 and the compared results in Figure 2.

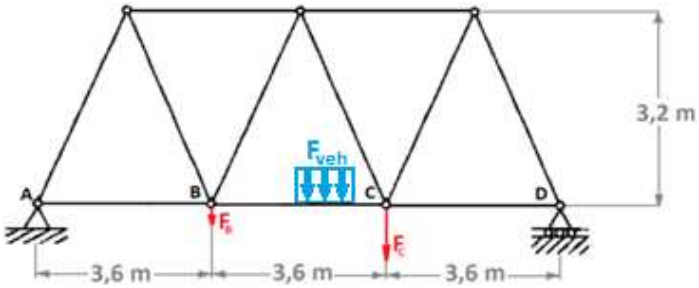

Fig. 1. Numerical example 1

Table 1

Execution time of FEM algorithm with different data types representation

data type format	pre-calculations time	loop-calculations time
double	0.067 s	0.868 s
float	0.063 s	0.822 s
int	0.086 s	0.548 s

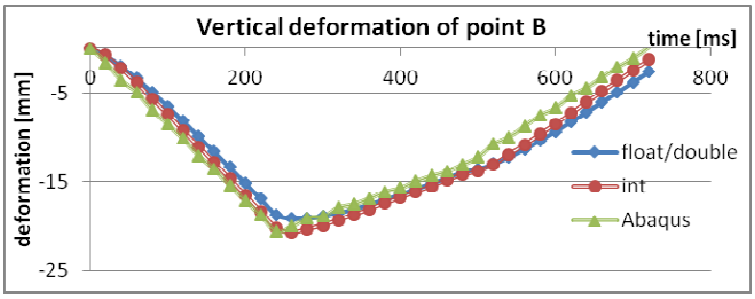

Fig. 2. Numerical example 1 results for different variants of the algorithm

By changing double-precision to a single-precision format, the computation time has been reduced by 5.7% with results nearly identical (the relative difference about 0.1%). By representing every floating-point variable and constant with integer format, the improvement of 28.3% has been obtained with only a little decrease of accuracy (the average difference between results in *float/double* variant and Abaqus results was 0.40 mm and between *int* variant and Abaqus results was 0.69 mm, the relative difference between *float/double* and *int* variant about 6% while the absolute difference never exceeded 1.5 mm). The ARM-CortexM4F architecture is only equipped in single-precision FPU and 64-bit register bank supports double format. The user has no influence on how the computations are performed.

The second example concerns the same truss as above, but with a different load case. A Heaviside load (800 kN, Fig. 3) has been applied to point B (in vertical direction). The results of vertical deformation of point B are presented in Figure 4.

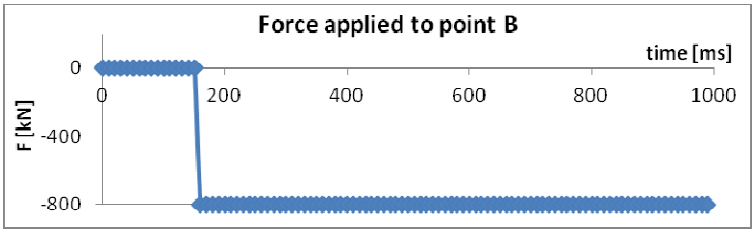

Fig. 3. Heaviside load

By changing the variables and constants representation from a double-precision format to an integer format, execution time was reduced from 2.14 to 1.54 s. 28.3% improvement has been obtained while maintaining high accuracy (as shown in Figure 4, the relative difference between *double* and *int* variant was about 5%).

The obtained results were also compared with the same finite element model solution from Abaqus software - implicit scheme was utilized, with the same time step 10 ms. As one can see in Figure 4, the results from Abaqus and microcontroller differ a little from each other, especially at the end of the simulation. The author suspects that it is related to numerical damping implemented in Abaqus. In Abaqus

the operators used for implicit direct time integration introduce some artificial damping in addition to Rayleigh damping.

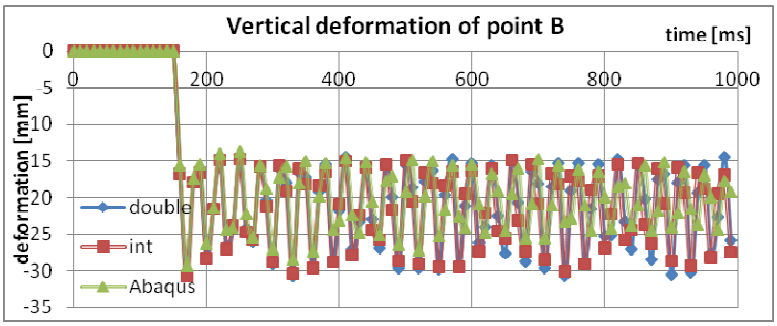

Fig. 4. Numerical example 2 results for different variants of the algorithm

4. Conclusions

It is crucial to optimize the FEM algorithms considering computation time, because in real time applications every millisecond matters. The presented idea of changing the representation of floating-point variables and constants to integer format gives very good results if the scaling is done correctly. This can be implemented to describe HIL simulations using RTFEM.

The paper proves that FEM can be successfully implemented to algorithms executed on ARM microcontrollers. By introducing improvements described in the paper, the execution time is significantly shortened, while maintaining high accuracy.

The presented examples may be simple, however the computations are performed on a microcontroller, a device that has been typically utilized for simpler tasks (control, communication, data processing). However new technologies, drawing along a significant increase of computational possibilities, open a wide range of new possibilities for applications of microcontrollers.

References

[1] Vigneron L.M., Verly J.G., Warfield S.K., Modelling Surgical Cuts, Retractions, and Resections via Extended Finite Element Method, MICCAI, 2004, LNCS 3217, Springer-Verlag, Berlin, Heidelberg 2004, 311-318.

[2] Lapeer R.J., Gasson P.D., Karri V., Simulating plastic surgery: From human skin tensile tests, through hyperelastic finite element models to real-time haptics, Progress in Biophysics and Molecular Biology 2010, 103, 208-216.

[3] Berkley J., Turkiyyah G., Berg D., Ganter M., Weghorst S., Real-time finite element modeling for surgery simulation: An application to virtual suturing, IEEE Transactions on Visualization and Computer Graphics 2004, 10(3), 314-325.

[4] Isobe Y., Watanabe H., Yamazaki N., XiaoWei L., Kobayashi Y., Miyashita T., Hashizume M., Fujie M.G., Real-Time Temperature Control System Based on the Finite Element Method for Liver Radiofrequency Ablation: Effect of the Time Interval on Control, Engineering in Medicine and Biology Society (EMBC), 35th Annual International Conference of the IEEE 2013, 392-396.

[5] Shin E., Real-time recovery of impact force based on finite element analysis, Computers and Structures 2000, 76, 621-635.

[6] Jaramillo A.E., Boulanger P., Prieto F., On-line 3-D system for the inspection of deformable parts, Int. J. Adv. Manuf. Technol. 2011, 57, 1053-1063.

[7] Duriez C., Control of elastic soft robots based on real-time finite element method, Robotics and Automation (ICRA) 2013, 3982-3987.

[8] Mucha W., Real-time hybrid simulation using materials testing machine and FEM, Advances in mechanics: theoretical, computational and interdisciplinary issues, Proceedings of the 3rd Polish Congress of Mechanics (PCM) and 21st International Conference on Computer Methods in Mechanics (CMM) - PCM-CMM-2015, Gdańsk, Poland, 8-11 September 2015, CRC Press/Balkema, 2016, 419-422.

[9] Mucha W., Kuś W., Application of mode superposition to hybrid simulation using Real Time Finite Element Method, Mechanika, Kauno Technologijos Universitetas, accepted, 2016.

[10] Mrozek A., Kuś W., Sztangret Ł., Real-time evolutionary optimization of metallurgical precesses using ARM microcontroller, Computer Methods in Material Science (CMMS) 2016, 16(1), 20-26

[11] Kuś W., Burczyński T., Bioinspired algorithms in multiscale optimization, Advanced Structured Materials 2010, 1, 186-192.

[12] Kokot G., Orantek P., The topology optimization using evolutionary algorithms, Solid Mechanics and its Applications 2004, 117, 173-186.

[13] Ogierman W., Kokot G., A study on fiber orientation influence on the mechanical response of a short fiber composite structure, Acta Mechanica 2016, 227(1), 173-183.

[14] Zienkiewicz O.C., Taylor R.L., The Finite Element Method, Volume 1: The Basis, Fifth Edition, Butterworth-Heinemann, 2000.

[15] Majchrzak E., Mochnacki B., Metody numeryczne. Podstawy teoretyczne, aspekty praktyczne i algorytmy, Wydawnictwo Politechniki Śląskiej, Gliwice 2004.

[16] Chandrupatla T.R., Belegundu A.D., Introduction to Finite Elements In Engineering, Fourth Edition, Pearson, Upper Siddle River, 2012.

[17] Golub G.H., Van Loan C.F., Matrix Computations, Third Edition, The Johns Hopkins University Press, 1996.

[18] Saad Y., Iterative Methods for Sparse Linear Systems, Second Edition, Society for Industrial and Applied Mathematics, 2003.

[19] Press W.H., Teukolsky S., Vetterling W., Flannery B., Numerical Recepies, The Art of Scientific Computing, Third Edition, Cambridge University Press, 2007.

[20] Yiu J., The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors, Third Edition, Elsevier, 2014.