Proceedings of the Fifteenth International Conference on
Computational Structures Technology
Edited by: P. Iványi, J. Kruis and B.H.V. Topping
Civil-Comp Conferences, Volume 9, Paper 14.1
Civil-Comp Press, Edinburgh, United Kingdom, 2024
ISSN: 2753-3239, doi: 10.4203/ccc.9.14.1

Hardware Accelerated Python Based Finite Element Analysis of Reinforced Concrete Member

H. Chung and H.-G. Kwak

Department of Civil and Environmental Engineering, KAIST
Daejeon, Republic of Korea

Abstract

This paper presents a nonlinear finite element analysis (FEA) of RC beams by adopting material models based on machine learning (ML). The Gaussian Process Regression (GPR) approach is considered for constructing concrete and steel material models. However, a GPR material model has an increased computational load, so it is difficult to use in the nonlinear analysis of RC structures composed of numerous members. To solve this limitation, GPU acceleration is based on the constitution of the parallelized computing structure. GPU-accelerated Python-based FEA program is developed to trace the nonlinear behaviour of RC beams. The FEA and experimental data for two representative RC beams are compared. The results obtained from the developed program confirm that the solution procedure using GPU acceleration with GPR material models can effectively be used in the nonlinear analysis of large RC structures with nonlinear behaviours.

Keywords: Hardware acceleration, Finite Element Analysis, Gaussian process, Reinforced Concrete, Parallel processing, High performance computing.

1 Introduction

Modern numerical simulation methods such as finite element analysis (FEA) and the boundary element method (BEM) have been popularly adopted in many different fields to identify physical phenomena, and the structural analysis field is no exception. Since structural behavior, particularly nonlinear behavior, is based on conservation and material laws, the prescribed stress-strain relation needed to be defined even if

the amount of corresponding experimental data is very limited. In recent years, however, unlike in the conventional approaches, a material model has been directly defined from data due to the application of the machine learning (ML) technique [1], [2] which can resolve the material uncertainty that was represented by the random parameters in the conventional approach and has the flexibility for the supplementation of additional experimental data [3].

In addition to the adoption of ML techniques, the data processing in the training and inference procedure also needs to be considered because, as mentioned above, implementation of ML usually requires a large amount of computational load and, results in a huge increase in total time consumption even over simple structural members [4]. As a result, the graphics processing unit (GPU) is popularly used for training and testing with most of the ML algorithms, including the GPR model. Since the GPU computing ecosystem that is based on a scalable parallel architecture has a large number of computing cores in a single unit, unlike the central processing units (CPU), which have a very limited number of cores, a parallelized computing structure is important when considering the GPU acceleration. This type of accelerating operation using the GPU as a co-processor of the CPU is called a general-purpose GPU (GPGPU) and is easily implemented in the Compute Unified Device Architecture (CUDA) platform. However, hardware acceleration is not a simple handover to another processing component. Since the GPU cannot process the given task alone as it is a co-processor, communication between the CPU and GPU acts as an overhead and significantly decreases the efficiency of computing performance if the program requires frequent communication between processors. That means that the overall tasks need to be carefully constructed to minimize communication while the task maintains the vectorized structure. Furthermore, a limited amount of dedicated memory space in the GPU imposes restrictions on the applicability of hardware acceleration, so using memory-efficient variables, such as sparse matrices, is important and raises the limit of the maximum possible dataset size of hardware acceleration [5]. Finally, due to the architecture difference, GPU cannot handle all of the instruction set CPU can handle, and mostly the operation that GPU can handle is currently limited to the linear algebra operations.

Of ML techniques, Gaussian Process Regression (GPR) has been widely used for estimating the compressive strength of the material [6], for composing a constitutive material model [7], and for defining a data-driven material model for stochastic structural analysis [8]. However, this data-driven method is mostly based on stochastic inference, which leads to a drastic increase in the computational cost compared to conventional constitutive material models, which are largely based on arithmetic operations, because the stochastic inference is based on the calculation of the probabilistic field over higher dimensions. Regarding this high cost of computing, an optimization and computing strategy for the material nonlinear analysis with the GPR material model was needed.

Upon constructing the GPR-based material models for concrete and reinforcement based on synthetic monotonic uniaxial stress-strain relationships formed from the idealized constitutive models, we developed the data-driven Python-based FEA program to trace the nonlinear behavior of reinforced concrete (RC) beams. We used the Timoshenko beam theory [9]. Then, the entire implicit static layered beam solution

procedure, from constructing the element stiffness matrix to the internal force determination, was vectorized, optimized for modern computers, accelerated with the GPU, and reduced memory pressure with sparse matrices. We compared the obtained numerical results from the implicit nonlinear FEA of simply supported RC beams using the constitutive and GPR material models to verify if the ML material model could be effectively used. Moreover, to verify the efficiency of the proposed solution procedures, comparisons with and without acceleration were performed to demonstrate the necessity for the adoption of a GPU in analyzing the nonlinear behavior of large structures discretized with numerous finite elements with a complicated material model, and then the entire implicit static layered beam solution procedure from the construction of the element stiffness matrix to the internal force determination was vectorized, optimized for modern computers, accelerated with the GPU, and reduced the memory pressure with sparse matrices. We compared the obtained numerical results from the implicit nonlinear FEA of simply supported RC beams using the constitutive and GPR material models to verify if the ML material model could be effectively used. Moreover, to verify the efficiency of the proposed solution procedures, comparisons with and without acceleration were performed to demonstrate the necessity for adopting a GPU in analyzing the nonlinear behavior of large structures discretized with numerous finite elements with a complicated material model.

2 Methods

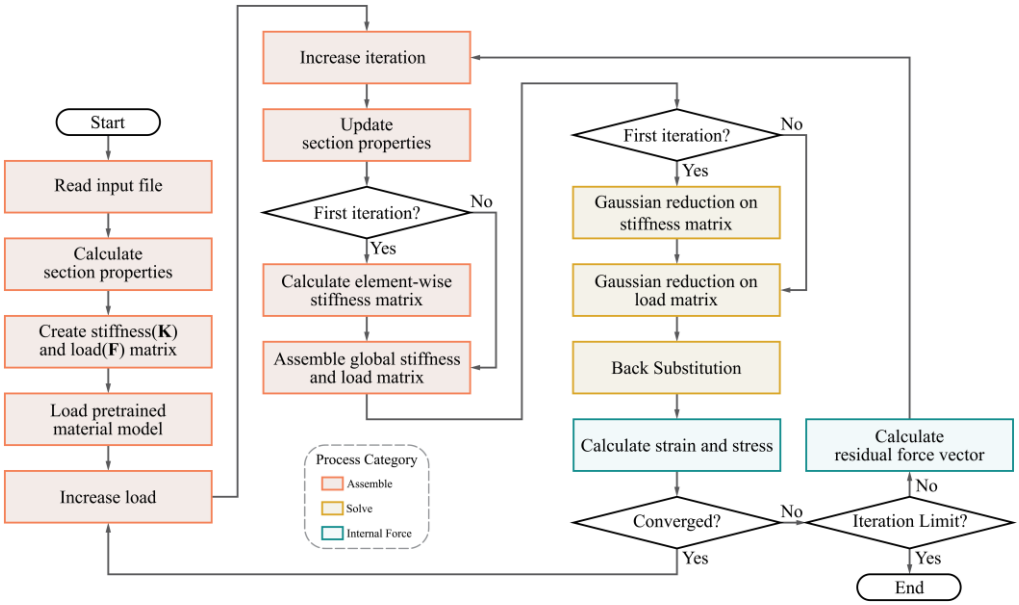


Figure 1: Conventional nonlinear analysis flowchart

To exactly analyze the improvement in the subprocess performance, we divided the whole solution procedure into three major categories, as shown in Figure 1. The first one is the ‘assemble’ category, which covers the solution procedure from reading the input file to the construction of the global stiffness matrix and global load vector. The

second one is the ‘solve’ category, in which solving the constructed equilibrium equation is performed and the nodal displacements are determined. Since this solving subprocess accounts for most of the solution procedure, it can be one of the significant bottleneck points from the time consumption aspect. Performing the Gaussian elimination [10] on a $n\ x\ n$ matrix requires time complexity of $O(n^3)$, which means bit complexity will grow exponentially [11]. The last category is the ‘internal force’ category, where the stresses and strains corresponding to the determined deformations are calculated. Despite relatively simple operations, re-evaluating the neutral axis in a section with the trial-and-error method causes another bottleneck in computation time. Beyond the three major categories, the evaluation of residual forces and additional iterations are followed up to converge results in the nonlinear analysis. We designed an improved approach to reduce computation time for these solution procedures. Different from a typical Gaussian elimination procedure which performs the time-consuming processes of updating the global stiffness matrix into an upper diagonal matrix and the succeeding back substitution sequence in each iteration to obtain the solution of the equation, the designed approach only determines the reduction factors from Gaussian elimination once and is used consistently in the following iterations unless the stiffness remains the same. Only the one-dimensional load vector is updated without updating the global stiffness matrix in every iteration procedure. Then, back substitution over the same stiffness matrix with an updated load vector is performed, which means the reduction factors determined from the previous iteration can be used again, and the amount of computing is drastically reduced. The computational cost-saving effect comes from not updating the global stiffness matrix, which is enlarged with an increase in the size of the stiffness matrix. However, despite these efforts, this heavily sequential structure is unsuitable for GPU acceleration, and vectorizing revision is necessary.

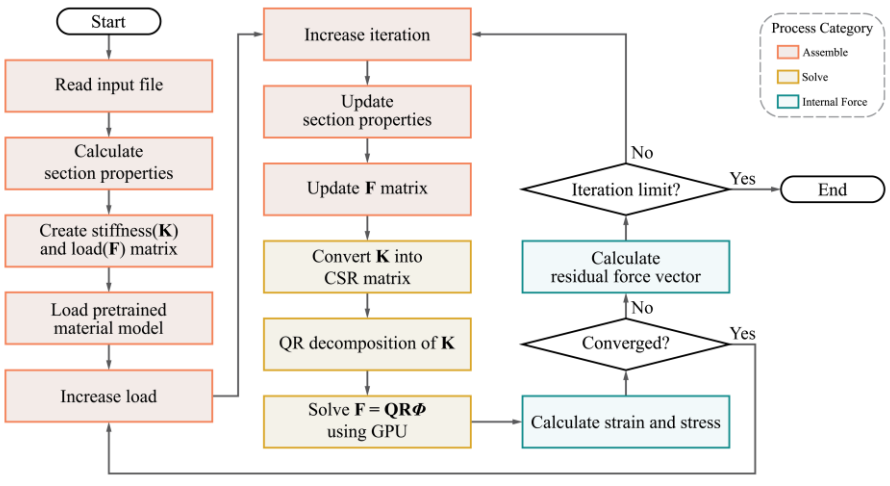


Figure 2: Vectorized nonlinear analysis flowchart

The overall configuration for the vectorized code structure is described in Figure 2. As can be seen by comparing it with Figure 1, the major change is made in the 'solve' category to maximize the efficiency of performance in the solution procedure. Instead of following the conventional Gaussian elimination procedure, the solution procedure is reconstructed with the use of the sparse solver, which is based on the cuSOLVER's CSRLSVQR routine, which is based on QR decomposition which is numerically more stable than the direct matrix inverse. Unfortunately, however, the cuSOLVER is proprietary software, so the source code for that library is not available. The CSR sparse matrix significantly reduces the operation required for the solution procedure. Furthermore, the entire code is rewritten in CuPy and changed to optimize the procedure for the GPU, using the broadcast functionality and vectorizing technique. Using the GPU takes additional time to transfer the information from the CPU to the GPU and then return the result to the CPU again. In these processes, two processing units are synchronized and wait until the execution inside one of the processing units finishes. That means that even the latest and fastest GPU can cause unexpectedly worse performance. Accordingly, to overcome this problem and improve the evaluation efficiency, the vectorized program in this paper stores all the variables required to perform an analysis in dedicated GPU memory from the beginning, and the numerical operation is conducted with the GPU exclusively while keeping the CPU to only control the flow of the process.

Converting into the vectorized analysis process, the broadcasting technique, which is the key to vectorizing and saving memory space for array operations, is implemented because it can remove the loop in the sequential structure without making needless copies of data [12]. The use of the broadcasting technique makes it possible to automatically expand the variable into a higher dimensional data structure without any initialization of the memory space [12], [13], and it will be a significant advantage in analyzing RC beams in which an RC section is discretized with multiple elements and layers of concrete and steel. Even in the overall vectorized analysis process, however, the iteration process to increase the load level and to check the convergence still needs to be performed with the implicit nonlinear analysis. Since the other bottleneck in the analysis procedure is on the 'internal force' category for the calculation of the stresses and strains, the related subprocess needs to be optimized for the GPU implementation because the GPR model used in this study was based on the PyTorch library which can seamlessly communicate between CuPy libraries using the DLPack.

In these solution procedures, the biggest difference in the modified solution procedure is in evaluating internal stresses. Unlike the conventional constitutive material models, the adoption of the machine learning material model drastically increases computing time, and repeated queries in finding the stress corresponding the considered strain will increase the total time consumption. Since much computing time is induced from moving the data back and forth between the CPU and GPU and waiting for the devices to synchronize rather than calculating the prediction value itself, the process to determine the neutral axis is modified for the GPU to resolve this bottleneck in the solution procedure. Owing to KeOps[14], loading the training data into the GPU does not require a lot of memory space, and quite a lot of test data can also be sent to the GPU. That means that many possible neutral axis candidates in an

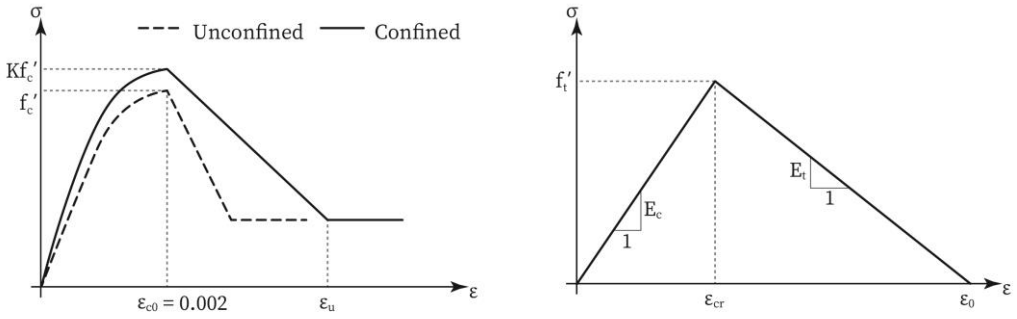
RC section can be prepared and queried simultaneously. Then the strains corresponding to each neutral axis are calculated, and the stresses corresponding to the strains can be queried at once. Since the neutral axis is the base used to determine the strains at each layer in an element, the calculation number of strains will increase in proportion to the number of possible neutral axis candidates considered while increasing the size of the corresponding stress matrix. All the matrix operations related to an increase and/or decrease in matrix size are performed on the basis of the broadcasting technique [13] by passing the appropriate indexing information into a lower dimensional matrix. A simple explanation for the difference between the two methods is that the conventional method evaluates only a single neutral axis candidate at each iteration, but the revised method evaluates multiple neutral axes at each iteration.

Beam	$E_c\ (MPa)$	$E_s\ (MPa)$	$f_c\ (MPa)$	$f_y\ (MPa)$	$\rho\ (A_s/bd)$
T1MA	26,632.8	194,571	31.7343	371.382	0.0062
J4	26,201.2	203,404	33.3426	309.596	0.0099

Note: E_c and E_s=the modulus of elasticity of concrete and steel, f_c=the compressive strength of concrete, f_y=the yield strength of steel

Table 1: Material properties for T1MA and J4 beam

To analyze the impact of adopting ML based material model, simple GPR based monotonic uniaxial stress – strain material model for concrete and steel is defined. The constitution of a GPR-based material model requires the generation of a synthetic experimental dataset because the real experimental dataset for the considered structure was not sufficiently available. If many real or related experimental datasets for the considered material are secured, then a synthetic experimental dataset does not need to be generated. Without exception, however, since no supplementary experiments to determine the stress-strain relations of concrete and steel which were used in the select beams were performed, we generated synthetic experimental datasets in this study based on the conventional stress-strain relations of the modified Kent and Park model for concrete [15] and the simple bilinear model which has been proven to be accurate enough by multiple studies on steel [16], [17]. These models are the most representative relations popularly used in nonlinear analyses and are presented in Figure 3 and Figure 4. More details for the expression of the stress-strain relations can be found elsewhere [15], [16], [18].



(a) (b)

Figure 3: Uniaxial concrete stress – strain material model

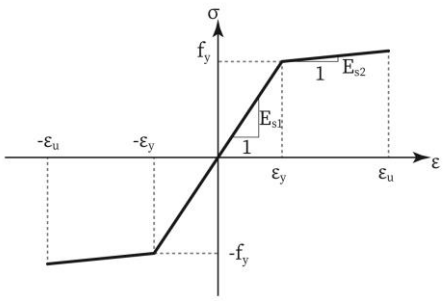
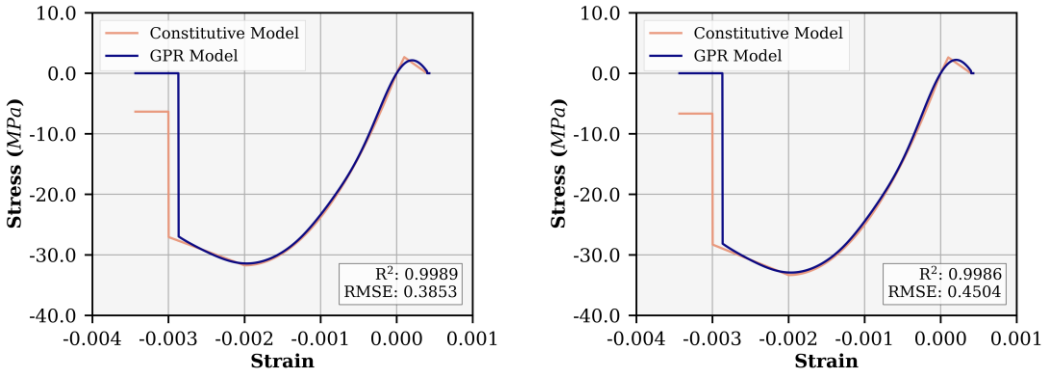


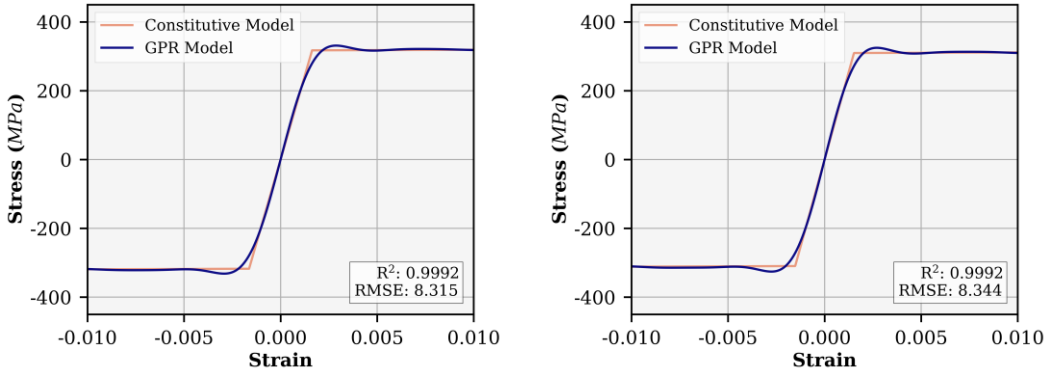
Figure 4: Uniaxial steel stress – strain material model

In constructing GPR-based stress-strain relations of concrete and steel, the material properties of RC beams of T1MA [19], [20], and J4 [21], which were analyzed in this study, need to be defined. The material properties given in Table 1 were directly determined from the experiments and served as the major parameters in constructing the stress-strain relations, which means that these major parameters for the constitutive model were randomly adjusted during the dataset generation for the GPR model. The random distribution followed the Gaussian distribution with the given parameters as the average values and one-tenth of the mean value as the standard deviation. Because of insufficient experimental data, furthermore, we also assumed the given parameters were independent of each other, and the value of standard deviation was arbitrary. The primary reason for the construction of a dataset for each parameter was to maximize the probability of the GPR model corresponding to the constitutive model. Thirty cases of the stress-strain relation composed of 200 data points were generated for each material, and the number of these data points, which can deliver an accurate description of the stress-strain relation, was determined through several repeated trials with a change in the number of cases and the number of data points. One of the main reasons for generating this limited amount of synthetic data is due to the limitation of the memory of the GPU. Since the Gaussian process regressor requires the training data to make a prediction, available memory spaces for both training data and test data need to be prepared in the prediction phase of the nonlinear finite element analysis process.



(a) (b)

Figure 5: Training result for concrete: (a) T1MA, (b) J4



(a) T1MA (b) J4

Figure 6 Training result for steel: (a) T1MA, (b) J4

Beam	ε_{c0}	$f_c'(MPa)$	ε_{cr}	$f_t'(MPa)$	ε_y	$f_y\,(MPa)$
T1MA	-1.972x10^{-3}	-31.42	2.049 x10^{-4}	2.139	2.872 x10^{-3}	331.1
J4	-1.968 x10^{-3}	-32.93	2.049 x10^{-4}	2.208	2.713 x10^{-3}	324.7

Table 1**:** Metrics in GPR material model

Figure 5 and Figure 6 represent the GPR concrete and steel stress-strain relations constructed on the basis of the mean value, and the obtained major metrics are listed in Table 2Table 1. Due to the prediction structure of the GPR material model to minimize the computational load, there is some difference in predicting the ultimate strain $\boldsymbol{\varepsilon_{cu}}$, as shown in Figure 5. Different from the constitutive concrete model in Figure 3 which assumes the maintenance of the resistance even after the ultimate strain is reached, the machine learning model in Figure 5 assumes that concrete completely loses its strength at the ultimate strain. In conclusion, the GPR concrete and steel stress-strain relations almost coincide with the constitutive models in the entire strain range, despite a slight difference in the compressive and tensile strength of concrete and the yield strength of steel. The calculation of the quality metric for both concrete and steel is limited to the specific bound of the region of interest. For the concrete, R^2 was 0.7859 for T1MA and 0.7864 for J4, and RMSE was 5.490 for T1MA and 5.751 for J4. It may seem very poor quality compared to a typical criterion, but this large difference between the models came from the stress response after the ultimate strain. If the calculation is limited to the non-zero stress area, the quality metric rises up to the point where the GPR model is exchangeable with the constitutive model; the value is noted in Figure 5.

3 Results

To compare the efficiency of adopting the machine learning material models of concrete and steel upon the GPU-based solution procedure, we selected two RC beams, T1MA [19] and J4 [21], because these RC beams have been popularly chosen in the classical nonlinear analyses of RC structures to verify the exactness of the introduced numerical models. We describe the different material properties of a concrete matrix and embedded reinforcements based on a layer model that divides an RC section into a finite number of imaginary layers. We considered 18 layers and

assigned the third layer to steel reinforcement. Figure 7 represents the configurations and loading conditions for T1MA and J4, respectively. Since it is well known that the average crack spacing is generally three times the concrete cover thickness [22] and an element length of less than the crack spacing does not improve the numerical result anymore, despite an increase in the number of elements [22], the maximum number of beam elements used in the numerical analyses is limited to 3840 with equal length. To compare the calculation time spent in the nonlinear analysis with the change in the number of elements, the two specimens are discretized with 60 to 3840 elements, double the number of elements.

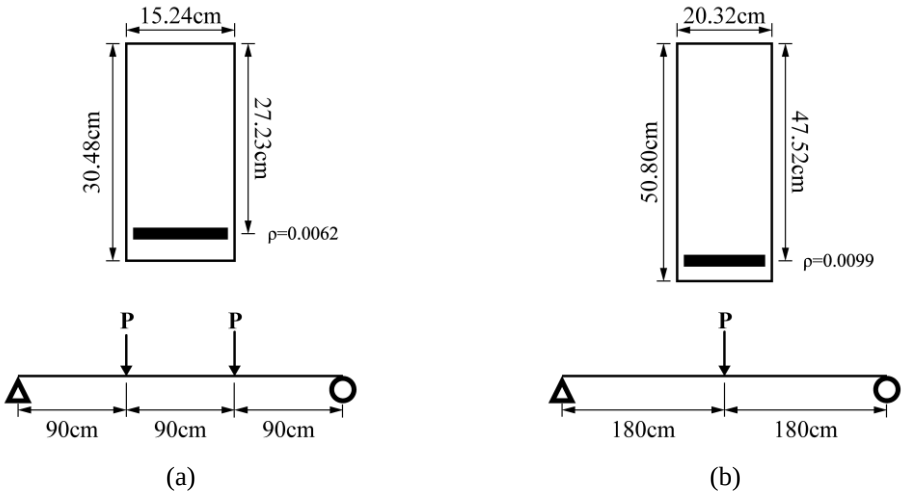
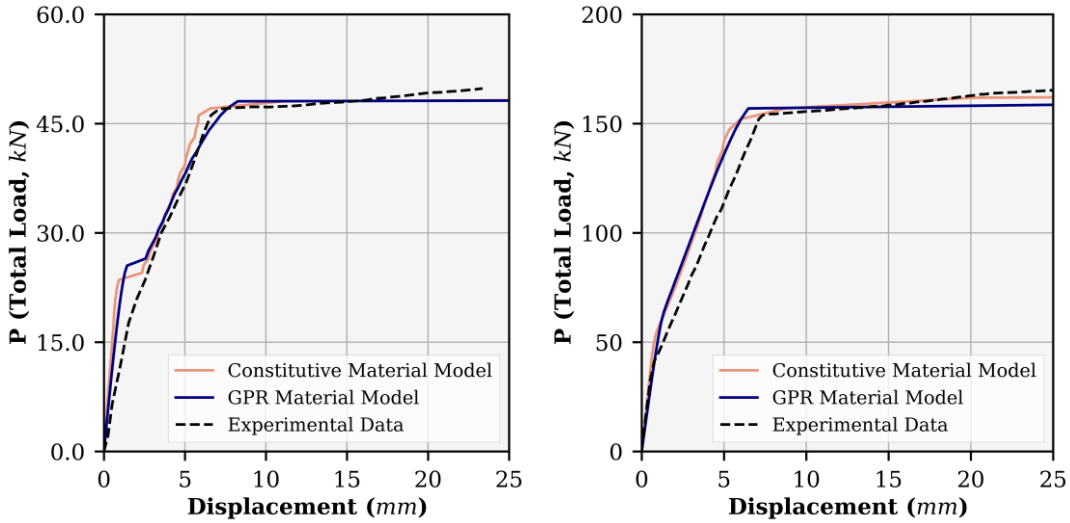


Figure 7: Specifications of beams: (a) T1MA beam, (b) J4 beam

Figure 8 compares the numerical results and the experimental data at the midspan of the RC beams. As shown in this figure, both material models, the machine learning material model, and the constitutive material model, effectively trace the RC beams' nonlinear behavior from the concrete cracking to the yielding of the steel. The initial discrepancy between the analysis and the experiment in T1MA stems from the fact that the specimen was probably extensively cracked before loading or the very small elastic displacement was inaccurately measured because the initial slope in the experimental data shows a difference from the elastic behavior before cracking. A slightly larger estimation of the yield strength using the machine learning material model was induced from the derived stress-strain relations of steel in Figure 6 which include the upper and lower yield strengths. Since these specimens are under-reinforced, the structural responses, including the yield strength, are dominantly governed by the stress-strain relation of steel. T1MA is more dominantly affected by the yield strength of steel because one-third of the center in the span will broadly be subjected to steel yielding. Even though the use of the machine learning material models of concrete and steel results in a 2.28% higher yield load and a 1.55% lower ultimate load from the experimental data, as shown in Figure 8(a), the entire prediction for the nonlinear behavior of the two specimens is still satisfactory.



(a) (b)

Figure 8: Load-Displacement Diagram of Beams with Two Different Material Models: (a) T1MA, (b) J4

In parallel with the adoption of the machine learning material model, we also implemented the GPU-accelerated vectorized solution procedure and examined its effectiveness through a comparison of the time taken in the solution procedure. Calculation of the total time was based on the average of three times because the computing time can be affected by the ambient temperature [23], as the heat dissipating performance of the computing system is affected by the ambient temperature and the total load is divided by 12 and 15 for T1MA and J4, respectively. Figure 9 presents the total time spent by the processing unit. The benchmark case (CASE A) is based on the vectorized CPU-based solution procedure with the conventional constitutive material model. Since the comparison of CASE A with CASE B shows that only the adoption of the GPR material model without GPU acceleration drastically increases the total time consumption, we can infer that this large jump of total time consumption comes from the increase in the computational cost of the GPR material model compared to the constitutive material model. In Figure 9, a sudden increase in time in the GPU processing (CASE B) is due to the memory allocation unit size configuration inside the program, which can degrade or improve the performance degradation and is limited to the graphics hardware memory size and bandwidth. However, adopting the GPU acceleration with the GPR material model (CASE C) alleviates the total time consumption again. It represents almost the same time consumption as that of CASE A. The accelerated case, the revised solution procedure with the GPR material model with GPU acceleration, clearly shows a performance improvement over the case without acceleration. CASE C is also expected to be more effective than CASE A in the nonlinear structural analyses of large structures discretized with multitudinous finite elements (see the crossing point between CASE A and CASE C in Figure 9). Moreover, as can be observed in Figure 9, the deviation of total time consumption in CASE B ranges from 29.91 to 1081.71, but it is reduced to a range from 0.58 to 42.02 in CASE C. The significantly lower deviation leads to the applicability of the accelerated solution procedure to the

structural analysis with many elements and a large degree of freedom without worrying about the uncertainty of an unexpectedly longer time consumption of the program. Overall, the accelerated solution procedure had 2.93 to 6.08 times faster processing speed than the case without an acceleration solution procedure, and we concluded that the adoption of the GPR material model must be based on the use of the GPU acceleration.

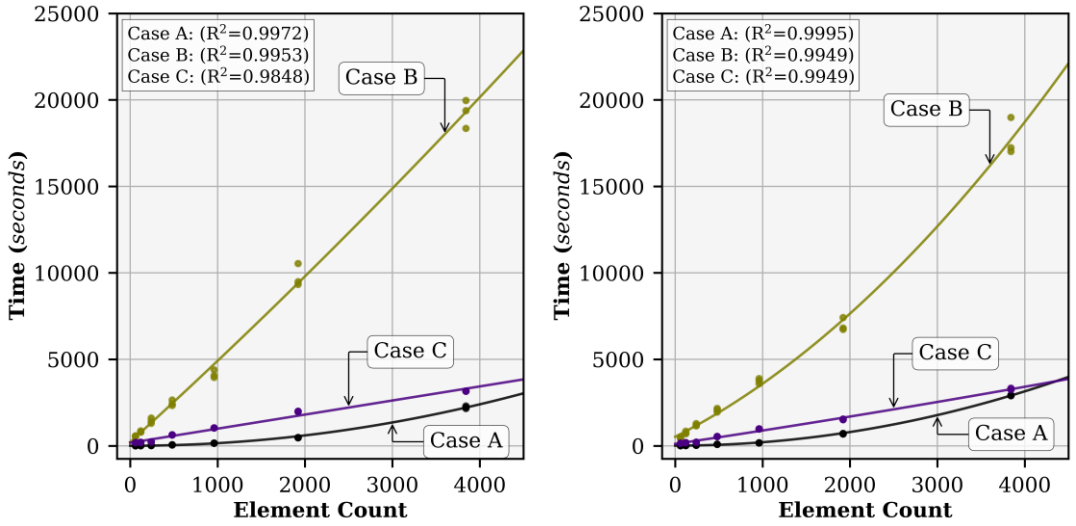


Figure 9: Total time consumption by processing Unit: (a) T1MA, (b) J4

Case A: Conventional constitutive material model without GPU acceleration
Case B: GPR material model without GPU acceleration
Case C: GPR material model with GPU acceleration

4 Conclusions and Contributions

This paper introduced an improved solution procedure that can effectively be used to trace the nonlinear behavior of large RC structures composed of numerous RC members. Use of the GPR-based material model can compensate for the roughness in the conventional constitutive material models induced from a limited amount of experimental data and still have the flexibility for supplementation of additional experimental data, but an increase in computational cost and an increase in the total time consumption of numerical analysis is the drawback of the GPR based material model and make it difficult to apply it to large RC structures. We implemented the graphics processing unit (GPU) acceleration to overcome the limitation of the GPR-based material model and updated the solution procedure to optimize the GPU process by adopting vectorizing and broadcasting while maintaining the entire analysis process inside the GPU. All the introduced solution procedures were integrated and finally represented as the data-driven Python-based GPU-accelerated FEA program.

Acknowledgements

This work was supported by the Korea Government as the National Research Foundation of Korea (NRF) grant (No. RS-2023-00220921) and the Korea Ministry of Land, Infrastructure and Transport (MOLIT) as an “Innovative Talent Education Program for Smart City”.

References

[1] S. Freitag, W. Graf, and M. Kaliske, “FE Analysis using Recurrent Neural Networks for Uncertain Stress-Strain-Time Dependencies,” *PAMM*, vol. 11, no. 1, pp. 211–212, Dec. 2011, doi: 10.1002/PAMM.201110097.

[2] W. Zhao, J. K. Liu, and Y. Y. Chen, “Material behavior modeling with multi-output support vector regression,” *Appl Math Model*, vol. 39, no. 17, pp. 5216–5229, Sep. 2015, doi: 10.1016/J.APM.2015.03.036.

[3] B. Chen, L. Shen, and H. Zhang, “Gaussian Process Regression-Based Material Model for Stochastic Structural Analysis,” *ASCE ASME J Risk Uncertain Eng Syst A Civ Eng*, vol. 7, no. 3, Sep. 2021, doi: 10.1061/AJRUA6.0001138.

[4] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, “Efficient Machine Learning for Big Data: A Review,” *Big Data Research*, vol. 2, no. 3, pp. 87–93, Sep. 2015, doi: 10.1016/J.BDR.2015.04.001.

[5] B. Charlier, J. Feydy, J. A. Glaunès, F. D. Collin, and G. Durif, “Kernel Operations on the GPU, with Autodiff, without Memory Overflows,” *Journal of Machine Learning Research*, vol. 22, pp. 1–6, Mar. 2020, doi: 10.48550/arxiv.2004.11127.

[6] N. D. Hoang, A. D. Pham, Q. L. Nguyen, and Q. N. Pham, “Estimating Compressive Strength of High Performance Concrete with Gaussian Process Regression Model,” *Advances in Civil Engineering*, vol. 2016, 2016, doi: 10.1155/2016/2861380.

[7] J. Wang, T. Li, F. Cui, C. Y. Hui, J. Yeo, and A. T. Zehnder, “Metamodeling of constitutive model using Gaussian process machine learning,” *J Mech Phys Solids*, vol. 154, p. 104532, Sep. 2021, doi: 10.1016/J.JMPS.2021.104532.

[8] B. Chen, “Gaussian Process Regression-Based Data-Driven Material Models for Stochastic Structural Analysis,” The University of Sydney, Sydny, 2022. [Online]. Available: https://hdl.handle.net/2123/28827

[9] D. R. J. Owen and E. Hinton, *Finite Elements in Plasticity : Theory and Practice*. Swansea, United Kingdom: Pineridge Press, 1980.

[10] R. W. Farebrother, *Linear Least Squares Computations*. Routledge, 1988.

[11] X. G. Fang and G. Havas, “On the Worst-case Complexity of Integer Gaussian Elimination,” *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC*, pp. 28–31, 1997, doi: 10.1145/258726.258740.

[12] “Broadcasting — NumPy v1.23 Manual.” Accessed: Dec. 04, 2022. [Online]. Available: https://numpy.org/doc/1.23/user/basics.broadcasting.html

[13] “Broadcasting Semantics — PyTorch 1.12 documentation.” Accessed: Dec. 04, 2022. [Online]. Available: https://pytorch.org/docs/1.12/notes/broadcasting.html
[14] “Block-sparse reductions — KeOps.” Accessed: Dec. 04, 2022. [Online]. Available: https://www.kernel-operations.io/keops/python/sparsity.html
[15] D. C. Kent and R. Park, “Flexural Members with Confined Concrete,” *Journal of the Structural Division*, vol. 97, no. 7, pp. 1969–1990, Jul. 1971, doi: 10.1061/JSDEAG.0002957.
[16] D. Ngo and A. C. Scordelis, “Finite Element Analysis of Reinforced Concrete Beams,” *ACI Journal Proceedings*, vol. 64, no. 3, pp. 152–163, 1967, doi: 10.14359/7551.
[17] A. Vebo and A. Ghali, “Moment-Curvature Relation of Reinforced Concrete Slabs,” *Journal of the Structural Division*, vol. 103, no. 3, pp. 515–531, Mar. 1977, doi: 10.1061/JSDEAG.0004579.
[18] B. D. Scott, R. Park, and M. J. N. Priestley, “Stress-Strain Behavior of Concrete Confined by Overlapping Hoops at Low and High Strain Rates,” *ACI Journal Proceedings*, vol. 79, no. 1, pp. 13–27, 1982, doi: 10.14359/10875.
[19] J. R. Gaston, C. P. Siess, and N. M. Newmark, “An Investigation of the Load-Deformation Characteristics of Reinforced Concrete Beams Up to the Point of Failure,” Urbana, Illinois, 1952.
[20] J. R. Gaston, C. P. Siess, and N. M. Newmark, “A layered finite element non-linear analysis of reinforced concrete plates and shells,” *Civil engineering studies, SRS*, vol. 389, 1972.
[21] N. H. Burns and C. P. Siess, “Load-Deformation Characteristics of Beam-Column Connections of Reinforced Concrete,” *Civil engineering studies, SRS*, vol. 234, 1962.
[22] W.-F. Chen, *Plasticity in Reinforced Concrete*. New York: McGraw-Hill, 1982.
[23] A. Mutapcic, S. Boyd, S. Murali, D. Atienza, G. De Micheli, and R. Gupta, “Processor Speed Control with Thermal Constraints,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, pp. 1994–2008, 2009, doi: 10.1109/TCSI.2008.2011589.