A Meshless Solution of a Small-Strain Plasticity Problem

Filip Strniša[a], Mitja Jančič[a,b], Gregor Kosec[a]
[a] Jožef Stefan Institute, Parallel and Distributed Systems Laboratory, Ljubljana, Slovenia
[b] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
filip.strnisa@ijs.si, mitja.jancic@ijs.si, gregor.kosec@ijs.si

***Abstract*—When the deformations of a solid body are sufficiently large, parts of the body undergo a permanent deformation commonly refereed to as plastic deformation. Several plasticity models describing such phenomenon have been proposed, e.g. von Mises, Tresca, etc. Traditionally, the finite element method (FEM) is the numerical tool of choice for engineers who are solving such problems. In this work, however, we present the implementation of the von Mises plasticity model with non-linear isotropic hardening in our in-house developed MEDUSA library, utilizing a variant of meshless methods – namely the radial basis function-generated finite differences (RBF-FD). We define a simple plane stress case, where a 2D block is fixed at one edge, and a tensile force, which causes the block to deform, is applied to it at the opposite edge. We show that results are in good agreement with the numerical solution obtained by Abaqus FEA, a commercial FEM solver.**

***Index Terms*—plasticity; meshless; radial basis function-generated finite differences**

I. Introduction

Speaking very broadly, a deformation of a solid body can be broken down into two main sub-categories: elastic and plastic deformation. It is said, that the deformation is elastic, if the body returns to its original shape after the applied load had been released, while plastic deformation occurs when any part of a solid body undergoes a non-reversible change of shape due to sufficiently large load applied [1]. Generally, the material response beyond the elastic-plastic tipping point, commonly referred to as *yielding criterion*, is non-linear [2], thus numerical treatment of partial differential equations is required [2].

Traditionally, such problems are solved with the finite elements method (FEM) [3]–[6]. In this work, however, we employ meshless methods that have proven to be a good alternative as they can operate on nodes contrary to mesh-based methods that require meshes [7]. An often used variant of the meshless methods is the radial basis function-generated finite differences (RBF-FD) [8], which has already been employed to obtain solutions to elasticity [9], [10] and plasticity [11], [12] problems.

We present our implementation of a von Mises plasticity model with non-linear isotropic hardening limited to small strains in a plane stress example. The implementation was done using our in-house developed MEDUSA C++ library [13] supporting all the required meshless procedures. The original FEM formulation of the solution procedure provided by the de Souza et al. [2] is adapted to employ RBF-FD and used to solve a simple plane stress problem.

The authors would like to acknowledge the financial support of Slovenian Research Agency (ARRS) in the framework of the research core funding No. P2-0095 and the World Federation of Scientists.

II. Plasticity

Plasticity problems, where an external force acts on a solid body, are usually solved by applying partial loads to the system i.e. the external force is applied incrementally [2], [14]. At each increment of the external force one first predictably solves the steady-state Navier-Cauchy equation for elastic bodies, and corrects the solution if the plastic yield criterion is violated. This section will present the small-strain plasticity model for plane stress cases. The steady state Navier-Cauchy equation can thus be written with Lamé constants as:

$$\left(\frac{2\lambda\mu}{\lambda+2\mu}+\mu\right)\nabla(\nabla\cdot\boldsymbol{u})+\mu\nabla^2\boldsymbol{u}=-\boldsymbol{r}, \qquad (1)$$

where $\boldsymbol{u}=(u_x,u_y)$ is the displacement vector. $\lambda=\frac{E\nu}{(1-2\nu)(1+\nu)}$ and $\mu=\frac{E}{2(1+\nu)}$ are the Lamé parameters, which depend on Young's modulus E, and Poisson's ratio ν. $\boldsymbol{r}$ is the residual force density. The latter is computed as the difference between internal and external force densities: $\boldsymbol{r}=\boldsymbol{f}^i-\boldsymbol{f}^e$. Internal force density is defined as:

$$\boldsymbol{f}^i=\nabla\cdot\sigma, \qquad (2)$$

where σ is the stress tensor with components: σ_{xx}, σ_{yy}, and $\sigma_{xy}=\sigma_{yx}$. Other components are assumed to equal 0 when assuming plane stress. For simplicity σ can be reshaped into a vector $\sigma=(\sigma_{xx},\sigma_{yy},\sigma_{yx})$. σ is related to $\boldsymbol{u}$ *via* the elastic strain tensor ε^E and the elastic stiffness tensor D as $\sigma=\mathrm{D}\varepsilon^E$. ε^E can be reshaped as $\varepsilon^E=(\varepsilon^E_{xx},\varepsilon^E_{yy},2\varepsilon^E_{yx})$, and then D becomes:

$$\mathrm{D}=\begin{pmatrix}2\mu+\frac{2\lambda\mu}{\lambda+2\mu} & \frac{2\lambda\mu}{\lambda+2\mu} & 0\\ \frac{2\lambda\mu}{\lambda+2\mu} & 2\mu+\frac{2\lambda\mu}{\lambda+2\mu} & 0\\ 0 & 0 & \mu\end{pmatrix}. \qquad (3)$$

In plasticity ε^E is a part of total strain tensor $\varepsilon=(\varepsilon_{xx},\varepsilon_{yy},2\varepsilon_{yx})=\varepsilon^E+\varepsilon^P$, where $\varepsilon^P=(\varepsilon^P_{xx},\varepsilon^P_{yy},2\varepsilon^P_{yx})$ is the irreversible plastic strain. ε is computed as:

$$\varepsilon=\frac{\nabla\boldsymbol{u}+(\nabla\boldsymbol{u})^\intercal}{2}. \qquad (4)$$

As the force is increased incrementally by $\delta \boldsymbol{f}^e$, Eq. (1) is used to compute the displacement increment $\delta \boldsymbol{u}$. Thus after n increments $\boldsymbol{u}^n = \boldsymbol{u}^{n-1} + \delta \boldsymbol{u}$. After finding $\boldsymbol{u}^n$ with the elastic prediction, one computes σ, and checks the yield criterion, in this case the von Mises criterion Φ [2]:

$$\Phi = \frac{1}{2}\sigma^{\intercal}\mathrm{P}\sigma - \frac{1}{3}\sigma_Y^2(\varepsilon_{eq}^P) = \frac{1}{2}\xi - \frac{1}{3}\sigma_Y^2(\varepsilon_{eq}^P), \quad (5)$$

where ε_{eq}^P is the scalar, "equivalent" plastic strain, and σ_Y is the likewise scalar yield stress, and is defined by the yield function. It can be noted, that von Mises stress is $\sigma_{VM} = \sqrt{\frac{3}{2}\sigma^{\intercal}\mathrm{P}\sigma}$. P is defined as:

$$\mathrm{P} = \frac{1}{3}\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 6 \end{pmatrix}. \quad (6)$$

Where the yield criterion is violated ($\Phi > 0$), σ is then corrected *via* a local Newton-Rhapson method. Let there be a correction factor $\delta\gamma$. First it is set to $\delta\gamma = 0$, then the solver updates it with:

$$\delta\gamma^{new} = \delta\gamma^{old} - \frac{\Phi}{\Phi'}. \quad (7)$$

$\Phi' = \frac{1}{2}\xi' - \frac{1}{3}H'$ is the derivative of the yield criterion, where:

$$\xi' = -\frac{(\sigma_{xx}+\sigma_{yy})^2}{9\left(1+\frac{E\delta\gamma}{3(1-\nu)}\right)^3}\frac{E}{1-\nu} - 2\mu\frac{(\sigma_{yy}-\sigma_{xx})^2 + 4\sigma_{xy}^2}{(1+2\mu\delta\gamma)^3}, \quad (8)$$

$$H' = 2\sqrt{\frac{2}{3}}H\left[\sqrt{\xi} + \frac{\delta\gamma\xi'}{2\sqrt{\xi}}\right]\sigma_Y\left(\varepsilon_{eq}^P + \delta\gamma\sqrt{2\frac{\xi}{3}}\right). \quad (9)$$

H is the slope of σ_Y at $\varepsilon_{eq}^P + \delta\gamma\sqrt{2\frac{\xi}{3}}$. Afterwards ξ is updated:

$$\xi = \frac{(\sigma_{xx}+\sigma_{yy})^2}{6\left(1+\frac{E\delta\gamma}{3(1-\nu)}\right)^2} + \frac{(\sigma_{yy}-\sigma_{xx})^2 + 4\sigma_{xy}^2}{2(1+2\mu\delta\gamma)^2}. \quad (10)$$

Φ (Eq. (5)) is also updated, and Eqs. (7)–(10) are iterated over until the yield criterion is satisfied ($|\Phi| < \mathrm{tolerance}$). This is the local iteration.

With computed $\delta\gamma$ the variables are updated as:

$$\begin{aligned} \sigma^{new} &= \mathrm{A}\sigma^{old}, \\ \varepsilon^E &= \mathrm{D}^{-1}\sigma^{new}, \\ \varepsilon_{eq}^{P\ new} &= \varepsilon_{eq}^{P\ old} + \delta\gamma\sqrt{2\frac{\xi}{3}}, \\ \varepsilon^{P\ new} &= \varepsilon^{P\ old} + \delta\gamma\mathrm{P}\sigma^{new}. \end{aligned} \quad (11)$$

A is a $\delta\gamma$ dependent tensor, defined as:

$$\mathrm{A} = \begin{pmatrix} \frac{1}{2}(a_1+a_2) & \frac{1}{2}(a_1-a_2) & 0 \\ \frac{1}{2}(a_1-a_2) & \frac{1}{2}(a_1+a_2) & 0 \\ 0 & 0 & a_2 \end{pmatrix}, \quad (12)$$

where $a_1 = \frac{1-\nu}{1-\nu+\frac{1}{3}E\delta\gamma}$, and $a_2 = \frac{1}{1+2\mu\delta\gamma}$. With variables updated, $\boldsymbol{f}^i$ is updated *via* Eq. (2), and used in Eq. (1) to recompute $\delta\boldsymbol{u}$. This is repeated until $|\boldsymbol{r}| < \mathrm{tolerance}$. This is the global iteration. Once the global iteration is completed, $\boldsymbol{f}^e$ is increased by $\delta\boldsymbol{f}^e$. The whole process is repeated until $\boldsymbol{f}^e$ equals the prescribed value.

Such approach, where D remains unchanged throughout the computation, requires many global iterations. This can be improved by exchanging the elastic stiffness tensor with the consistent tangent operator, which is updated by the local iteration [2], [14]. However, for simplicity this is not explored in this work.

III. RBF-FD APPROXIMATION OF DIFFERENTIAL OPERATORS

Often mentioned advantage of the mesh-free methods over the mesh-based methods is that they can operate on scattered nodes. This is particularly convenient when complex three-dimensional domains are being treated, because in such cases, automated mesh generation is even today impossible without a human interference. While there are also other advantages, such as direct control over the approximation order, it is important to know that in general, mesh-free methods are computationally more complex because larger support sizes are needed.

Since the emergence of mesh-free methods, many approximation variants have been proposed. The first mentions of RBF-FD reach back in 2000 with the introduction from Tolstykh [15]. Since then, the method has been thoroughly studied and applied to numerous real-life problems.

In the context of RBF-FD, a linear differential operator $\mathcal{L}$ (for example operators in the Navier-Cacuhy equation (1)) is approximated in node $\boldsymbol{x}_c$ over set of n nearby nodes

$$\widehat{\mathcal{L}u}(\boldsymbol{x}_c) = \sum_{i=1}^{n} w_i u(\boldsymbol{x}_i) \quad (13)$$

for an arbitrary function u. A set of n neighboring nodes is often also referred to as the *stencil* nodes or *support* nodes. The unknown weights $\boldsymbol{w}$ are obtained for a given set of radial basis functions (RBFs) ϕ centered at the stencil nodes of a central node $\boldsymbol{x}_c$

$$\phi(\boldsymbol{x}) = \phi(\|\boldsymbol{x} - \boldsymbol{x}_c\|). \quad (14)$$

The approximation (13) can then be written in a linear system

$$\underbrace{\begin{bmatrix} \phi_1(\boldsymbol{x}_1) & \cdots & \phi_n(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_n) & \cdots & \phi_n(\boldsymbol{x}_n) \end{bmatrix}}_{\boldsymbol{\Phi}} \underbrace{\begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}}_{\boldsymbol{w}} = \underbrace{\begin{bmatrix} (\mathcal{L}\phi_1(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \\ \vdots \\ (\mathcal{L}\phi_n(\boldsymbol{x})\big|_{\boldsymbol{x}=\boldsymbol{x}_c} \end{bmatrix}}_{\ell_\phi}. \quad (15)$$

While researchers in the early work on RBF-FD used infinitely smooth RBFs, such as Gaussians or Mul-

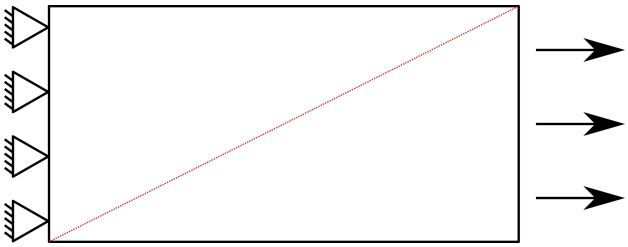


Figure 1. The sketch of the test case. The arrows on the east side represent the tensile stress. The red dashed line represents the results sampling line, used in later analyses. It runs from the south-west corner, to the north-east one.

tiquadrics, nowadays, piecewise smooth polyharmonic splines (PHS)

$$\phi(r) = \begin{cases} r^k, & k \text{ odd} \\ r^k \log r, & k \text{ even} \end{cases} \tag{16}$$

are commonly used. These, however, do no ensure the convergent behavior nor solvability of the system [8]. That is handled by additionally enforcing the constraint (13) for a set of $s = \binom{m+d}{d}$ monomials with up to and including degree m in a d-dimensional domain.

With the additional constraints, the approximation can be compactly written as

$$\begin{bmatrix} \boldsymbol{\Phi} & \boldsymbol{P} \\ \boldsymbol{P}^\mathsf{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\ell}_\phi \\ \boldsymbol{\ell}_p \end{bmatrix}, \tag{17}$$

where $\boldsymbol{P}$ is an $n \times s$ matrix of monomials evaluated at stencil points, $\boldsymbol{\ell}_p$ is the vector of values assembled by applying the considered operator $\mathcal{L}$ to the polynomials at $\boldsymbol{x}_c$, i.e $\ell_p^i = (\mathcal{L}p_i(\boldsymbol{x}))\big|_{\boldsymbol{x}=\boldsymbol{x}_c}$ and $\boldsymbol{\lambda}$ are Lagrangian multipliers. Finally, the weights $\boldsymbol{w}$ to approximate a differential operator $\mathcal{L}$ can be obtained by solving the system (17) using a standard solver from the Eigen library and Lagrangian multipliers are discarded.

IV. Example

For demonstration purposes, a simple plane stress problem is solved. We chose a 2D block (a rectangle), attached to a solid wall on the west edge ($\boldsymbol{u} = (0, 0)$) and pulled by a tensile stress on the east edge ($\sigma_{xx} = 30\,\text{MPa}$), with remaining edges being traction-free ($\boldsymbol{n} \cdot \boldsymbol{\sigma} = \mathbf{0}$, where $\boldsymbol{n}$ is the face normal), as is shown by the sketch in Fig. 1.

The tensile stress acts as the external force on the system, and is increased in steps of equal magnitude, as prescribed by the model input. Dimensions of the block are as follows: length $L = 10\,\text{mm}$, and height $H = 5\,\text{mm}$. Material properties in this case are chosen arbitrarily, and are: $E = 10\,\text{GPa}$, $\nu = 0.4$, and a yield function $\sigma_Y(\varepsilon_{eq}^P)$ defined by the following set of points $(\varepsilon_i^P, \sigma_{Yi})$: (0, 20 MPa), (0.001, 25 MPa), (0.005, 30 MPa), (0.02, 40 MPa) with a piecewise linear interpolation in-between.

The results of computations are analyzed along the red dashed line, which is also shown on the sketch in Fig. 1. As meshless discretization is used, the results along the line are interpolated using the Sheppard interpolation from Python's `photutils` package. To stabilize the computations, and to reduce computational time, the case

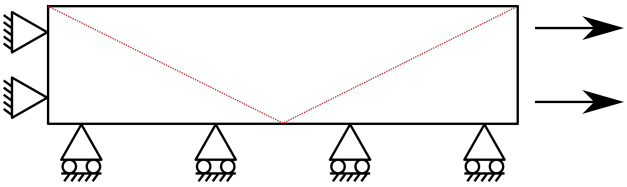


Figure 2. The sketch of the reduced case. See Fig. 1 for details. The sampling line is composed of two parts, and is reflected at $x = L/2$. The asymmetrical variables (u_y, and σ_{xy}) are transformed by the following function: $u(x) = -u,\ x < L/2; = u,\ x \geq L/2$.

TABLE I. Different discretization densities $\frac{dx}{L}$, and corresponding number of generated nodes in a domain.

$\frac{dx}{L}$ [−]	**No. nodes**
$\frac{1}{19}$	110
$\frac{1}{49}$	659
$\frac{1}{99}$	2585
$\frac{1}{149}$	5745
$\frac{1}{199}$	10204
$\frac{1}{249}$	15876
$\frac{1}{299}$	22852

is further simplified by reducing the domain size along the symmetry line ($u_y = 0$, $\frac{\partial u_x}{\partial y} = 0$), producing the problem setup to that shown in Fig. 2. To stabilize the computations even further, the corner nodes at Neumann boundary conditions (traction boundaries) are removed.

The computations are performed at different discretization resolutions, and different amount of load steps to display the model's convergence. Details on discretization resolutions are presented in Table I. Results are computed for each discretization, using $N_{\text{load}} \in \{5, 10, 50, 100\}$ load steps.

The entire implementation is written in C++, using our in-house developed MEDUSA library [13]. MEDUSA's built-in fill and relax algorithms [16] are used to create the uniform irregular computational domains. Differential operators are approximated with the RBF-FD and computed using the support of $n = 50$ nearest points. Eq. (1) is solved implicitly with the `BiCGSTAB` solver from the Eigen library for linear algebra [17]. In the main loop, the implicit solver operates sequentially, but local operations, such as computation of stress tensor σ, and the local iterations are executed in parallel with OpenMP. The compiling is done with the `g++` compiler, using the following flags: `-Wall -O3 -march=native -fopenmp -std=c++17`. The results were exported to text-VTK files, and processed with Python and ParaView. Computations were performed on a laptop, with an Intel Core i7-8750H CPU, and 16 GB DDR4 RAM.

For comparison and validation purposes, the same problem is also solved using the commercial software Abaqus FEA, which utilizes FEM. Abaqus solution is obtained with the full domain setup (Fig. 1) using $N_{\text{FE}} = 5000$

8-node biquadratic plane stress quadrilateral (CPS8R) finite elements resulting in total of 15301 nodes. Finite element type CPS8R was chosen to obtain a finite element with 4 integration points and thus reducing the error of mapping the secondary variables to nodal values and additionally avoiding the potential Hourglass and shear locking issues. Reduced integration mode additionally reduces the computational times.

V. Results

Example results of computations are shown in Figs. 3 and 4. Fig. 3 is displaying the σ_{xx} field on the discretized domain, with nodes displaced by $50\boldsymbol{u}$. A higher stress concentration can be observed in the north-western corner of the domain, which is surrounded by areas of lower σ_{xx} stress, which then homogenizes along the body toward the eastern edge. In Fig. 4 it can be observed, how the material begins to yield, as it deforms plastically – more drastic deformation increases can be observed after first couple of linear-elastic steps.

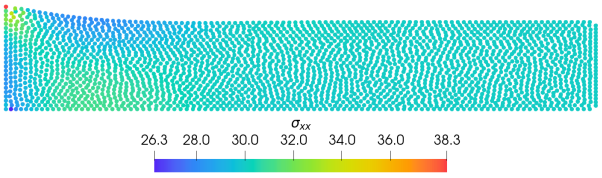


Figure 3. Visualization of the solution in a domain with 2585 nodes after 10/10 load steps. The nodes are displaced by $50\boldsymbol{u}$, and the color-map is showing σ_{xx}.

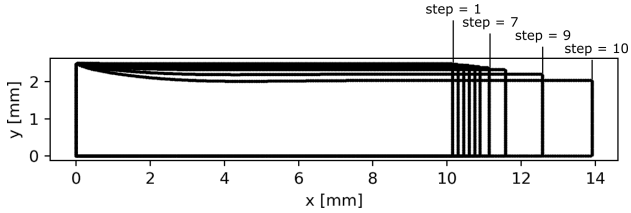
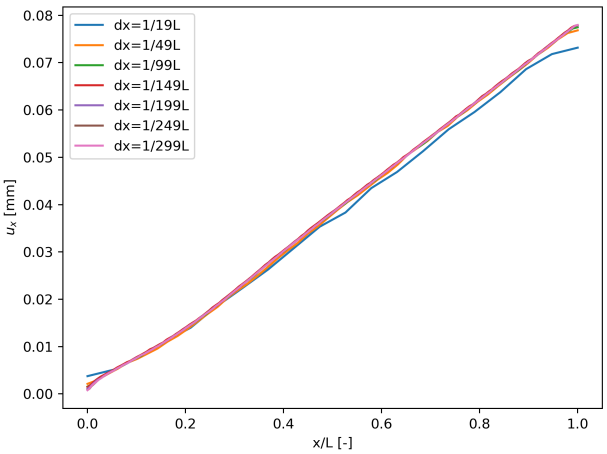


Figure 4. Visualization of the solution in a domain with 22852 nodes, showing the boundary nodes positions, displaced by $50\boldsymbol{u}$, at load steps 1/10 - 10/10.

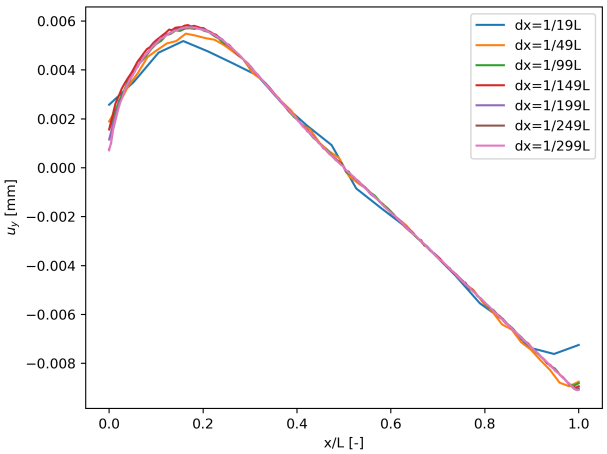
The model's convergence by increasing the node density can be seen in Fig. 5, and in Fig. 6 is showing the load-step dependent convergence for $\boldsymbol{u}$ at low (659 nodes) discretization density.

The solution with $N = 10204$ computational nodes and $N_{\text{steps}} = 100$ load steps is compared with the Abaqus FEA solution along the aforementioned diagonal. The comparison is made for solutions of $\boldsymbol{u}$ (Fig. 7), and σ (Fig. 8). In both cases the absolute difference plotted on the log scale represents $\Delta u = |u_{medusa} - u_{abaqus}|$, where indices "medusa" and "abaqus" are representing the meshless and Abaqus solutions, respectively.

Finally, all points from meshless (the same case as previously) and Abaqus solutions are plotted on the σ_{VM} against ε_{eq}^{P} graph along with the yield function in Fig. 9, to analyze how well they follow the yield function.



(a) u_x

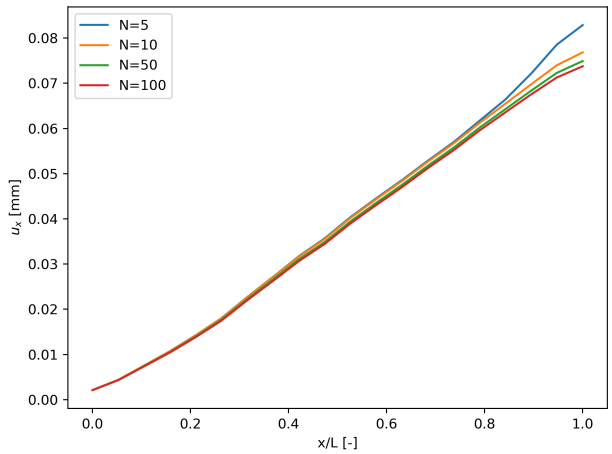


(b) u_y

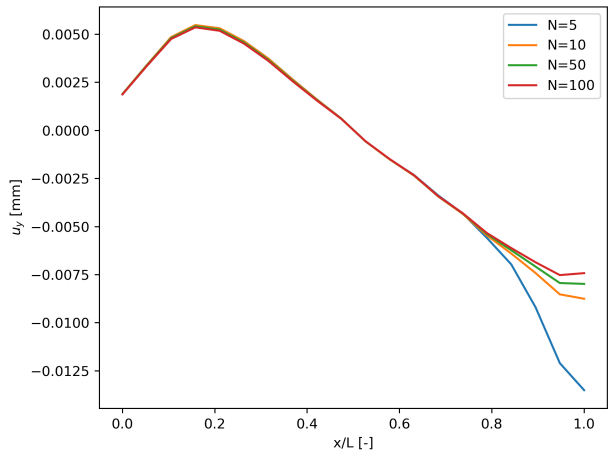
Figure 5. Solution convergence due to varying discretization density. u_x and u_y plotted against dimensionless coordinate x/L, $N_{\text{steps}} = 10$.

The convergence plots in Fig. 6 show that the quality of the solution may increase by increasing the number of load steps, but at higher density discretizations this may be insignificant. Finding the balance between discretization density and number of load steps for optimal accuracy and computational time is beyond the scope of this paper.

The comparison between the presented implementation of von Mises plasticity model with the commercial solver Abaqus (Figs. 7 and 8) shows good agreement between both solutions. This is shown by closely matched solution for $\boldsymbol{u}$ as is shown in Fig. 7, and a similar σ prediction in Fig. 8. One should note however, that the meshless domain had the corner nodes removed to stabilize the computation, which resulted in greater difference between the two solutions especially at 0 x-coordinate. The difference between the two solutions for σ decreases as one is going up the x-coordinate, except for the corner node, for reasons mentioned above. The decrease in the difference is likely due to σ_{xx} being prescribed at $x = L$ through the traction boundary condition, and both approaches fulfill it well. A source of error, which may contribute to the noisy

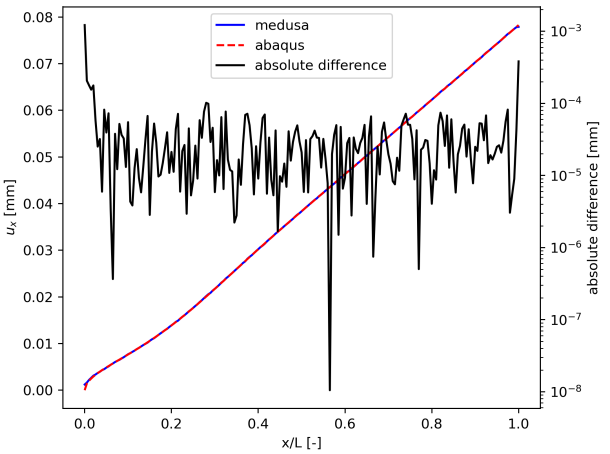


(a) u_x, 259 nodes.

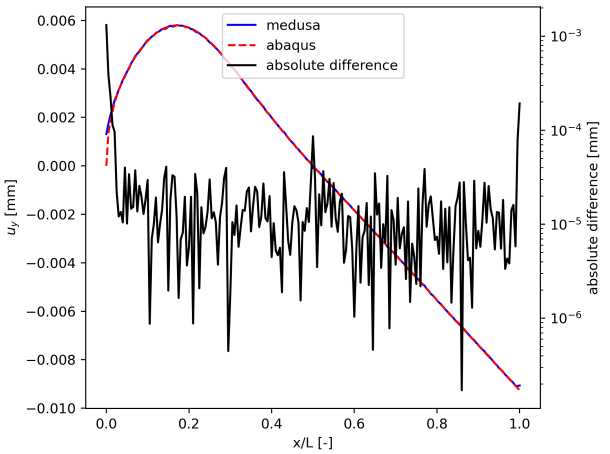


(b) u_y, 259 nodes

Figure 6. Solution convergence due to varying number of load steps N density. u_x and u_y plotted against dimensionless coordinate x/L, results for 259 nodes.



(a) u_x



(b) u_y

Figure 7. Comparison of both components of $\boldsymbol{u}$, and the absolute difference between Medusa and Abaqus results plotted against dimensionless coordinate x/L.

nature of the absolute difference plots, is the interpolation, which was used in post processing to extract the data along the prescribed lines.

Fig. 9 is showing that the implemented plasticity model closely follows the yield function. However, some of the values obtained by both approaches appear to be violating the yield criterion. In the meshless case these points are located on the fixed edge, and in Abaqus' case these are at both singularities at the fixed edge or in their close proximity.

VI. Conclusion

This work demonstrates the use of RBF-FD in modeling small-strain plasticity. Under the plane stress assumption, a simple 2D problem is solved, using the von Mises plasticity model. For comparison, an identical problem is also solved in Abaqus, which uses FEM. Results show good agreement between both approaches. The analyzed example case also indicates that increasing the number of load steps may stabilize the solution in a scarcely populated domain, whereas this effect is not as pronounced in a densely populated one. Generally it can be concluded that RBF-FD can substitute FEM for solving similar small-strain plasticity problems.

References

[1] Y. Fung and P. Tong, *Classical and Computational Solid Mechanics*, ser. Advanced series in engineering science. World Scientific, 2001. [Online]. Available: https://books.google.si/books?id=hmyiIiiv4FUC

[2] E. A. de Souza Neto, D. Peric, and D. R. J. Owen, *Computational Methods for Plasticity: Theory and Applications*. Wiley, 2008.

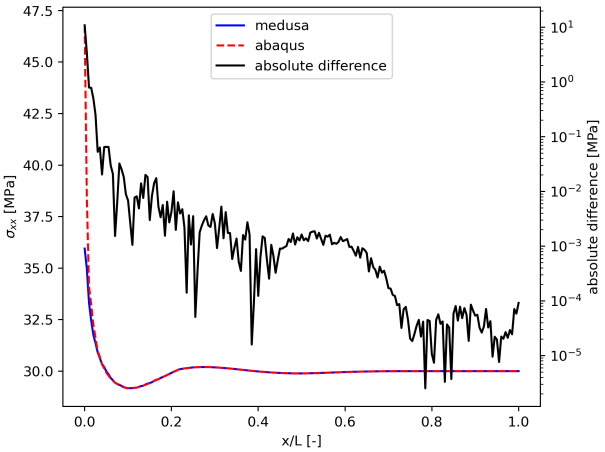
[3] S. Bartels, A. Mielke, and T. Roubíček, "Quasi-static small-strain plasticity in the limit of vanishing hardening and its numerical approximation," *SIAM J. Numer. Anal.*, vol. 50, pp. 951–976, 2012.

[4] B. Schröder and D. Kuhl, "Small strain plasticity: classical versus multifield formulation," *Archive of Applied Mechanics*, vol. 85, pp. 1127–1145, 2015.

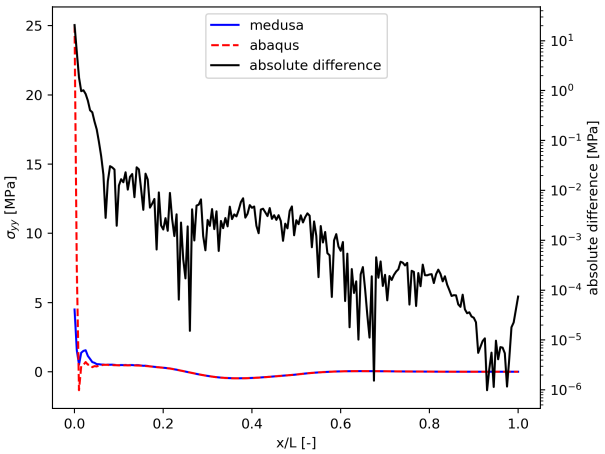
[5] A. A. Roostaei and H. Jahed, "A cyclic small-strain plasticity model for wrought mg alloys under multiaxial loading: Numerical implementation and validation," *International Journal of Mechanical Sciences*, 2018.

[6] G. Y. Amouzou and A. Soulaïmani, "Numerical algorithms for elastoplacity: Finite elements code development and implementation of the mohr–coulomb law," *Applied Sciences*, vol. 11, no. 10, 2021.

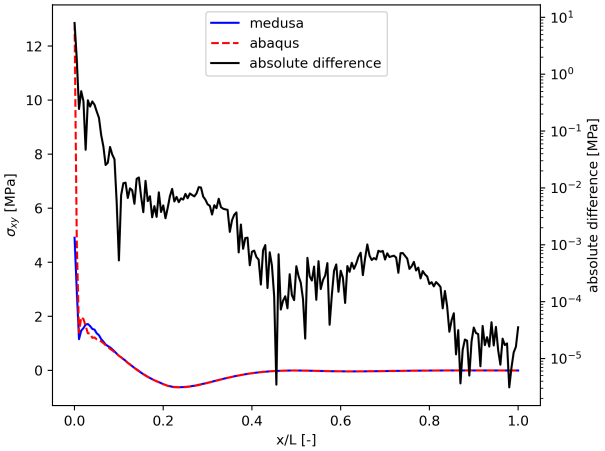
[7] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl, "Meshless methods: an overview and recent developments," *Computer methods in applied mechanics and engineering*, vol. 139, no. 1-4, pp. 3–47, 1996.



(a) σ_{xx}



(b) σ_{yy}



(c) σ_{xy}

Figure 8. Comparison of the three components of σ, and the absolute difference between Medusa and Abaqus results plotted against dimensionless coordinate x/L.

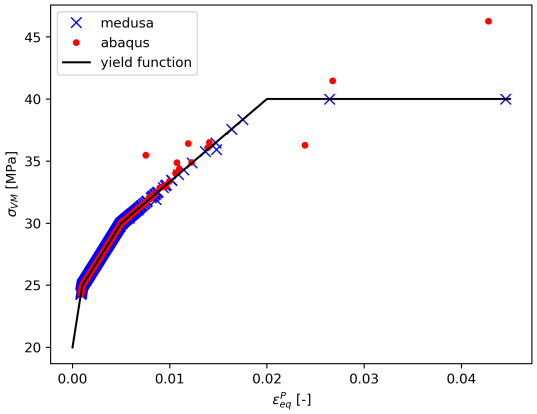


Figure 9. σ_{VM} plotted against ε^P_{eq} for meshless and Abaqus solutions. The solid line represents the yield function, which is flat beyond the (0.02, 40 MPa) point, resulting in perfect plastic behaviour.

[8] V. Bayona, N. Flyer, B. Fornberg, and G. A. Barnett, "On the role of polynomials in rbf-fd approximations: Ii. numerical solution of elliptic pdes," *Journal of Computational Physics*, vol. 332, pp. 257–273, 2017.

[9] M. Depolli and R. Trobec, "Computational efficiency of linear system construction for mlpg method on a multicore computer," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2019, pp. 200–205.

[10] J. Slak and G. Kosec, "Adaptive radial basis function-generated finite differences method for contact problems," *International Journal for Numerical Methods in Engineering*, vol. 119, no. 7, pp. 661–686, 2019.

[11] M. A. Jankowska, "On elastoplastic analysis of some plane stress problems with meshless methods and successive approximations method," *Engineering Analysis with Boundary Elements*, vol. 95, pp. 12–24, 2018.

[12] P. Jiang, H. Zheng, J. Xiong, and P. Wen, "Nonlinear elastic-plastic analysis of reinforced concrete column-steel beam connection by rbf-fd method," *Engineering Analysis with Boundary Elements*, vol. 128, pp. 188–194, 2021.

[13] J. Slak and G. Kosec, "Medusa: A C++ Library for solving PDEs using Strong Form Mesh-Free methods," *ACM Transactions on Mathematical Software*, 2021.

[14] V. M. Yarushina, M. Dabrowski, and Y. Y. Podladchikov, "An analytical benchmark with combined pressure and shear loading for elastoplastic numerical models," *Geochemistry, Geophysics, Geosystems*, vol. 11, no. 8, 2010.

[15] A. I. Tolstykh, "On using rbf-based differencing formulas for unstructured and mixed structured-unstructured grid calculations," in *Proceedings of the 16th IMACS world congress*, vol. 228. Lausanne, 2000, pp. 4606–4624.

[16] J. Slak and G. Kosec, "On generation of node distributions for meshless pde discretizations," *SIAM Journal on Scientific Computing*, vol. 41, no. 5, pp. A3202–A3229, 2019.

[17] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.