I. Software Features

This code performs parallel-based simulations of continuum damage mechanics problems. In its current formulation, the following features are available:

- The code models the performance of quasi-brittle materials, and the user can select between two relevant damage models that capture such a behavior: the Mazars [1] and the Geers [2] damage laws.
- The code performs quasi-static simulations for 2D problems (plane-strain or plane-stress).
- The code supports the analysis of both structured and unstructured meshes, with quadrilateral elements that contain 4 Gauss points.
- Two numerical solvers are available, namely the Newton-Raphson (NR) [3] and the Unified Arc-Length (UAL) [4] methods.
- The code supports displacement-driven analysis (Dirichlet BCs are imposed)
- The code can be run either in serial or parallel mode.

II. Software Prerequisites

This code requires **MATLAB** and the **Parallel Computing Toolbox** to be installed. We use MATLAB version R2022b. While we recommend using GMSH to generate a new mesh, the user can pick any relevant software of their choice. The generated mesh should be stored in a .txt file, modified by the user to match the format that is explained below (see the "IV. Model-files structure" section).

III. Code structure

The code is a compilation of several .m files. The main script is the "**FEM_Main_Script_2D.m**", which is the only code-related file the user needs to open and edit. In this file the user specifies the working/saving directories (line 13) and the model name (line 16). This file then calls the solver (Newton-Raphson or UAL) and the analysis starts.

Each file named "**func____.m**" serves a specific purpose, and a brief description of its functionality is provided at the beginning of the file. The overall code follows a typical FEM workflow (element-level procedures, assembly to global matrices, solution for unknown degrees of freedom).

The analysis generates output data at each load increment, which are stored in the "**Saved_Files**" folder. It also prints contour plots for damage, local strain, and nonlocal strain at the last load increment, which are stored in the "**Images**" folder.

IV. Model files structure

Each model is fully defined in two ".txt" files: a **mesh** file ("modelName_mesh.txt") and a **parameter** file ("modelName_parameters.txt"). The mesh file contains the definitions of:

- Nodal coordinates

- Element connectivity
- Imposed Dirichlet boundary conditions

The parameter file contains several hyperparameters of the problem, and they are presented in detail below (see Section "VIII. Details of parameter file"). The parameter file must reference the associated mesh file. This occurs at the last line of the parameter file, which links to the appropriate mesh file.

To facilitate the user's understanding of the model files, we provide in the "**Test Problems**" folder the files for 4 sample cases: a Symmetric Single Notch Tension (SSNT) problem with Coarse, Intermediate and Fine mesh resolution (following [5]), and a Double Notch Tension test (termed as Direct Tension Test in [6]). The user should create files that follow the structure of the provided benchmarks.

V. Running the code

In order for the user to implement our code on one of the provided benchmark problems (let us assume "SSNT_Coarse"), the following steps are required. The user needs to:

1. Ensure MATLAB and the Parallel Computing Toolbox are installed.
2. Download the "Code" and "Test Problems" folders.
3. Open the "SSNT_Coarse_parameter.txt" file and adjust the parameters according to their desired setup (see "VII. Details of parameter files" for details).
4. Ensure the correct mesh filename ("SSNT_Coarse_mesh.txt") is referenced at the last line of the parameter file.
5. Copy the mesh and parameter files to the "Code" folder.
6. Within the "Code" folder, create the "Saved Files" and "Images" folders.
7. Open the "FEM_Main_Script_2D.m" file and update the following:
    a. "main_file_path" with the directory of the "Code" folder (line 13)
    b. "model_name" with the mesh file to run (line 16)
8. Run the "FEM_Main_Script_2D.m"

VII. Details of parameter files

Here we summarize and briefly explain the primary parameters which are pertinent to the simulation, to facilitate the running and customization of each analysis.

1. *RoutineID*: specifies the non-linear solver. Values:
    a. 1 – for Unified ArcLength (UAL)
    b. 2 – for Newton-Raphson

2. *SolverID*: specifies whether a local or nonlocal damage theory is used. This variable must be consistent with the number of dofs/ per node, *No._DOF_per_node*. Values:
    a. 1 – for local damage (No._DOF_per_node: 2)
    b. 2 – for nonlocal gradient damage (No._DOF_per_node: 3)

3. *Scheme_ID*: this refers to the implementation scheme for UAL, according to [4]. Values:
    a. 1 – for Partitioned Consistent (PC) scheme
    b. 2 – for Partitioned Non-Consistent (PNC) scheme

4. *TangentID:* formulation of the Jacobian matrix. The current code supports the analytical version. Values:
    a. 1 – for analytical tangent

5. *Constraint_type*: For the UAL, it determines how the arc-length constraint is enforced. Values:
    a. 1 – for cylindrical (recommended)
    b. 2 – for spherical

6. *tolerance:* convergence criterion for the solver. For most continuum damage mechanics problems, a tolerance between 10^{-4} – 10^{-8} ensures a good accuracy-cost tradeoff.

7. *ST (strain tolerance):* additional tolerance specific to the UAL solver only, for controlling solution accuracy. Recommended range: 10^{-5} to 1e-7

8. *max_failed_attempts*: maximum allowed number of failed increments (UAL only).

9. *direction_load:* this is the direction where the load is applied. Values:
    a. 1 – for X axis
    b. 2 – for Y axis

10. *eq_strain_type:* this refers to the definition of the local equivalent strain. Values:
    a. 1 – for Mazars definition (principal strains)
    b. 2 – for shear strain
    c. 3 – for modified von Mises strain (in this case, *k_damage_parameter* is a control parameter relevant only here, with a recommended value is 10)

11. *Damage_type*: refers to the damage evolution law. Values:
    a. 1 – for Mazars damage law
    b. 2 – for Geers damage (only for nonlocal gradient damage)

12. *reaction_calc:* determines how boundary reactions are computed. Values:
    a. 1 – calculate reactions on homogeneous Dirichlet boundary condition [Default]
    b. 2 – calculate reactions on inhomogeneous (non-zero) Dirichlet boundary condition

VIII. Tips for testing and improved performance

- If using UAL, experiment with different Scheme_ID options (PC or PNC), strain tolerance (ST) and arclength limits to identify the approach which best suits the problem's convergence.

- For nonlocal gradient damage, the user must confirm that the associated length scale parameters and mesh refinements are appropriate to capture the damage gradient accurately.

- If the numerical solution doesn't converge, consider adjusting *max_failed_attempts*, *tolerance*, or *ST* values to allow for better convergence behavior.

- Always verify that boundary conditions are correctly imposed, particularly if reaction_calc = 2 (for non-zero Dirichlet boundaries).

- MATLAB parallelization can be run either as processes-based or thread-based. Our investigations have indicated that threads-based is optimal, so we have set it as the default in the code. To use the processes-based environment, the user should go to line 17 in the user inputs part of the "FEM_Main_Script_2D.m" file and replace 'Threads' with 'Processes'. The number of parallel threads to be used can be modified according to the available hardware capabilities of the user's device (default: 3) and can be changed in the following line.

IX. Contributions, Support, and Reporting Issues

The current code version can benefit from a series of user contributions, which are most welcome. These include:

- extension to other element types (i.e. higher-order elements),
- extension to other non-linear solvers (i.e. force-controlled arc-length),
- extension to other damage or phase-field models,
- extension to 3D problems,
- extension to multi-physics considerations, combining damage with other phenomena (i.e. heat transfer).

For contributions, support, or to report an issue, users can open an issue on GitHub.

References

[1] J. Mazars, (1986): "A description of micro-and macroscale damage of concrete structures", *Engineering Fracture Mechanics*, 25 (5-6), 729–737.

[2] M. Geers, R. De Borst, W. Brekelmans, R. Peerlings (1998): "Strain-based transient-gradient damage model for failure analyses", *Computer Methods in Applied Mechanics and Engineering* 160 (1-2) 133–153.

[3] T.J. Ypma (1995): "Historical development of the Newton-Raphson method", *SIAM Review*, 37(4), 531-551.

[4] R. P. Saji, P. Pantidis, M. E. Mobasher (2024): “A new unified arc-length method for damage mechanics problems”, *Computational Mechanics*, 1–32.

[5] P. Pantidis, M.E. Mobasher (2023): “Integrated Finite Element Neural Network (I-FENN) for non-local continuum damage mechanics”, *Computer Methods in Applied Mechanics and Engineering*, 404, 115766.

[6] R.H.J. Peerlings, R. de Borst, W.A.M. Brekelmans, M.G.D Geers (1998): “Gradient-enhances damage modelling of concrete fracture”, *Mechanics of Cohesive-Frictional Materials*, 3, 323-342.