Stand 7. Dezember 2020

Befehlsreferenz für DAEdalon

Programmablauf und Steuerung:

`dae`	Starten von DAEdalon (Starten der GUI).
`lprob`	Initialisierung einiger Variablen, Einlesen der Eingabefiles, Initialisierung der Historyfelder.
`stiffness`	Assemblierung der Systemmatrix $\boldsymbol{k}$ und des Residuumsvektors $\boldsymbol{r}$.
`syst`	Aufbau der globalen rechten Seite (aus Residuum und externen Lasten) sowie Einarbeitung der Verschiebungsrandbedingungen in $\boldsymbol{k}$ und $\boldsymbol{r}$.
`solv`	Lösen des linearen Gleichungssystems $\boldsymbol{k}\boldsymbol{\Delta u} = \boldsymbol{r}$.
`residuum`	Berechnung des Residuums $= \sqrt{\boldsymbol{\Delta u} \cdot \boldsymbol{\Delta u}}$.
`time`	Zeit um Δt imkrementieren ($\rightarrow$ Inkrementieren der Randverschiebungen, -lasten, da diese mit `tim` skaliert werden (sofern load_flag=1 gesetzt ist)). Erzeugen von Ausgabe- und Restartfiles sowie ausführen von Userscript falls die notwendigen Variablen zugewiesen sind. Das Zeitinkrement ist in `dt` gespeichert, die aktuelle Zeit in `tim`.

`go`	Ausführen der Befehle:
	`stiffness`
	`syst`
	`solv`
	`residuum`

`loop(XX)`	Führt XX Zeitschritte durch, innerhalb des Zeitschrittes werden maximal 15 Newton-Iterationen ausgeführt, sobald das Residuum $<$ 1.E–10 ist, wird die Zeit um `dt` inkrementiert und das neue GG ausiteriert.
`sf`	Schaltet den Sparse-Solver ein bzw. aus.
`lf`	Schaltet die Skalierung der Randlasten und Randverschiebungen mit der Zeit ein bzw. aus.
`mf`	Schalter zum Rausschreiben eines Movies in `movie_array`.
`opti`	Optimierung der Bandbreite durch Knotenneunummerierung.

Plotbefehle:

`mesh0`	Ausgabe undeformiertes Netz.
`meshx`	Ausgabe deformiertes Netz.
`disvec`	Ausgabe aller Verschiebungen als Vektoren.
`defo_scal=VALUE`	Verschiebungen werden mit `VALUE` skaliert dargestellt. → `meshx, disvec`
`nodenum`	Knotennummern dranschreiben.
`elnum`	Elementnummern dranschreiben.
`clearplot` bzw. `clp`	Löschen des Plotfensters, sollte auf jeden Fall vor dem ersten Plotten aufgerufen werden.
`reac`	Reaktionskräfte an Knoten mit vorgegbenen Randverschiebungen einzeichnen.
`boun`	Lager einzeichnen: grün: festgehaltene Verschiebungen. pink mit Pfeilen: vorgebene Randverschiebungen ungleich null.
`forc`	Vorgegebene Randlasten in blauen Pfeilen einzeichnen.
`dispx, dispy`	Ersetzt durch allgemeineren Befehl `ucont(X)`.
`cont(X)`	Contourplot der Größe X, die auf Elementebene in `cont_mat_gp` in Spalte X abgelegt ist. Normalerweise sind die ersten 6 Einträge folgende: `cont_mat_gp`(1:6)= $[\varepsilon_x, \varepsilon_y, 2\varepsilon_{xy}, \sigma_x, \sigma_y, \sigma_{xy}]$ In `cont_mat_node` befinden sich die auf die globalen Knoten projizierten GP-Werte.
`cont_sm(X1,X2)`	Wie cont(X), bei der Verwendung mehrerer Materialdatensätze wird Contourgröße X1 nur für Elemente aus Materialdatensatz X2 dargestellt.
`ucont(X)`	Contourplot der Verschiebungen (bzw. Knotenfreiheitsgrade) des Freiheitsgrades X.
`ucont_sm(X1,X2)`	Wie cont_sm(X1,X2), zur Darstellung der Verschiebungen (bzw. Knotenfreiheitsgrade) des Freiheitsgrades X1 für Elemente aus Materialdatensatz X2.
`mats`	Mehrere Materialdatensätze verschiedenfarbig darstellen.

Ausgabe von verschiedenen Größen:

`out('NAME')`	Schreibt das Ausgabefile `.\output\NAME_tim.out` (Knotengrößen: x, u, σ, ε, ...).
`histout('NAME')`	Schreibt das Ausgabefile `.\output\NAME_tim.out` (History-Feld an den GPs).
`out_file_name='NAME'`	Wird `out_file_name` ein Name zugewiesen, so wird in `time.m` automatisch `out('NAME')` aufgerufen.
`histout_file_name='NAME'`	Wie `out_file_name` aber für `histout('NAME')`.
`u2f2f('NAME')`	Schreibt die aktuellen Knotenverschiebungen ins File `./parser/NAME.f2f`, das Format ist so gewählt, dass die Datei direkt in ein FEAP-Eingabefile eingebaut und mit `f2f.pl` weiter verarbeitet werden kann.
`dis(KNOTENNUMMER)`	Ausgabe der Knotenkoordinaten und aller Freiheitsgrade von `KNOTENNUMMER`.

Anmerkung:
Soll nicht nach jedem Zeitschritt ein Output-File geschrieben werden, muss die Variable `out_incr` (Defaultwert=1) dementsprechend geändert werden. Durch sie wird festgelegt alle wieviel Zeitschritte ein Output-File geschrieben wird.

Restart-Files:

`rst_write('NAME')`	Schreibt das Restart-File `.\rst_files\NAME_tim.mat` (History-Feld an den GPs).
`rst_file_name='NAME'`	Wird `rst_file_name` ein Name zugewiesen, so wird in `time.m` automatisch `rst_write('NAME')` aufgerufen.
`restart('NAME')`	Rechnung durch Aufruf von Restart-File `NAME` vorsetzen.

Anmerkung:
Damit nicht nach jedem Zeitschritt ein Restart-File geschrieben wird, existiert die Variable `rst_incr` (Defaultwert=20). Durch sie wird festgelegt alle wieviel Zeitschritte ein Restart-File auch wirklich geschrieben wird.

Ausführen eines Userscriptes:

Durch setzen der Variable `userSkript='USERFILE.m'` wird das m-File `USERFILE.m` automatisch bei jedem Aufruf von time (also zu Beginn jedes Zeitschrittes außer wenn `tim=0`) aufgerufen.

GUI:

Die GUI startet normalerweise automatisch bei Aufruf von `dae.m`. Wird nach Eingabe von `dae.m` die GUI nicht gestartet, kann das im Plotfenster in der Menuleiste unter DAEcontrol manuell getan werden. Mit der dort festgelegten Einstellung wird DAEdalon auch beim Neuaufruf gestartet (mit GUI oder ohne GUI). Das automatische Öffnen der DAEdalon Homepage durch Klick auf DAEOnline funktioniert zur Zeit nur unter Linux.

Die GUI ist in fünf Hauptgruppen unterteilt, durch anklicken kommt man in die einzelnen Untermenues. Über die GUI können nicht nur die meisten DAEdalon-Funktionen gesteuert werden, sonderen auch das Preprocessing-Script `f2f.pl` und die Postprocessing-Scripte.

Achtung:
Durch Einführung der GUI hat sich die Befehlsfolge zum Einlesen der Inputfiles geändert. Durch Aufruf von `dae.m` wird DAEdalon gestartet und die GUI initialisiert. Um die Inputfiles einzulesen, muss `lprob.m` aufgerufen werden. Soll eine Rechnung mit neuem Inputfile gestartet werden, muss darum auch nicht mehr `dae.m` sondern nur noch `lprob.m` eingegeben werden.

Achtung 2:
Die GUI sollte unter Linux und unter Windows laufen, einziges Problem unter Linux ist, dass beim erstmaligen Aufrufen von `dae.m` die Menues nicht richtig erzeugt werden. nach dreimaligem eintippen von `dae.m` auf der Matlab-Konsole sollte aber alles funktionieren. Diese Prozedur muss nur beim erstmaligen Aufruf innerhalb der Matlab-Umgebung durchgeführt werden.

Achtung 3:
Wenn Fehler oder Probleme autauchen, bitte kurz eine email schicken.

Dynamik:

Die Implementierung folgt im wesentlichen dem Vorgehen in Wriggers und Hughes. Dämpfung wird durch Rayleigh-Dämpfung beschrieben (siehe Extradoku Dynamik). Alle für dynamische Berechnungen zusätzlich notwendigen Dateien (außer den Elementen) befinden sich in `./dynamics`. Alle bisher vorhandenen Materialgesetze können ohne Änderung verwendet werden, es ist jedoch darauf zu achten, dass im Eingabefile als letzte drei Parameter jedes Materialdatensatzes die Dichte des Materials und die beiden Rayleigh-Dämpfungskonstanten eingetragen sein müssen. Beispieleingabefiles sind `balken_dyn` und `balken3_dyn`.

`dyn_init`	Initialisierung der der benötigten Variablen, Ausschalten der Skalierung der Randlasten und -verschiebungen mit der Zeit (`load_flag = 0`).
`dyn_loop(XX)`	Durchführen von XX Zeitschritten, analog zu `loop(XX)`.

Bogenlängenverfahren:

Für lastgesteuerte Probleme bietet sich das Bogenlängenverfahen an, bei dem eine plastische Bogenlänge als Inkrement angegeben wird. Die Implementierung hält sich im Wesentlichen an das Vorgehen in Wriggers Seite 156-165, als Verfahren wurde das von Riks gewählt.
Verwendung:
`arcLoop(XX,YY)`
`XX`: Abbruchkriterium, wenn maximal zugelassenes Lastinkrement überschritten wird:
$\frac{P_{\text{aktuell}}}{P_0} \leq XX$ für $XX > 0$ und
$\frac{P_{\text{aktuell}}}{P_0} \geq XX$ für $XX < 0$
`YY`: Anzahl der maximal durchzuführenden Loops.