

Anleitung zum Konsolenprogramm "Lea2"

Erstellt: 2011-07-18



Stand: 2015-01-26

Inhaltsverzeichnis

| | | |
|---|---|----|
| 1 | Beschreibung..... | 3 |
| 2 | Aufrufparameter..... | 3 |
| 3 | Exit-Codes..... | 4 |
| 4 | Ausgabeformate..... | 5 |
| | LEAP-Format..... | 5 |
| | CSV-Ausgabe..... | 5 |
| | Hauptspannungen..... | 6 |
| | Interpretation des relativen Fehlers (Feld „relErr“..... | 7 |
| | Bedeutung der Fehlercodes (Feld „errCode“..... | 8 |
| | Funktionswerte-Ausgabe..... | 10 |
| 5 | Konfiguration..... | 10 |
| | Verfügbare Integrationsroutinen..... | 12 |
| | Beispiel für eine Konfigurationsdatei..... | 13 |
| | Beispiel für „LEAP-ähnliche“ Einstellungen..... | 13 |
| 6 | Format der Eingabedatei..... | 14 |
| | Formale Syntax..... | 14 |
| | Beispiel einer Eingabedatei..... | 15 |
| | Ausgabepunkte auf der Grenzfläche zwischen zwei Layern..... | 16 |
| 7 | Multithreading..... | 16 |
| 8 | Grenzen und Einschränkungen des Programms..... | 17 |
| | Literatur..... | 19 |

1 Beschreibung

Das Programm erwartet eine Liste von *Szenen*, jeweils bestehend aus einer *Struktur* (Schichtenmodell), einer Liste von *Ausgabepunkten* (Koordinaten) und einer Liste von *Lasten*.

Es berechnet die Verformungen, Spannungen und Dehnungen der Struktur unter dem Einfluss der Lasten an den angegebenen Ausgabepunkten.

2 Installation



isbs_path_lea2

3 Programmaufruf und Parameter

Der Start des Programms erfolgt über die Befehlskonsole:

```
>lea2 [Optionen...]
```

Parameter

Der Parameter **--help** liefert eine Übersicht der vom Programm verstandenen Aufrufparameter:

```
ISBS LEA2 Console Application (Version: 2013-12-29, s.reiser@tu-bs.de)
Usage: lea2 [OPTIONS]

-i, --infile: EINGABEDATEI. (Ohne Angabe: Lesen von STDIN.)
-o, --outfile: AUSGABEDATEI. (Ohne Angabe: Schreiben auf STDOUT.)
-t, --threads: ANZAHL THREADS. (Ohne Angabe: 1; 0 = Anzahl der CPUs benutzen.)
-c, --config: KONFIGURATIONSDATEI. (Ohne Angabe: Standardwerte verwenden.)
--save_config: KONFIGURATIONSDATEI. Standard-Konfigurationsdatei schreiben.
-f, --format: AUSGABEFORMAT. (Standardeinstellung = csv)
    csv
    CSV-Ausgabe
    integrands
    Keine Integration durchfuehren, Integranden als Wertetabelle ausgeben.
    matrix
    Ausgabe der Gleichungssystem-Matrix an der Stelle m=INTPOS. (Wert der
    (Integrationsvariable m mittels --intpos=INTPOS angeben. Default = 1.0)
-C, --read_config_from_input: Konfiguration kommt mit der Eingabedatei.
-P, --fixed_outputpoints: Zuerst def. Punkte auf alle Strukturen anwenden.
-L, --fixed_loads: Zuerst definierte Lasten fuer alle Strukturen verwenden.
--num_only: Rein numerische Integration ohne Sonderbehandlung des 1. Layers.
--no_comments: Keine mit # eingeleiteten Kommentare in Eingabe verwenden.
-h, --help: Diese Hilfe zeigen.
```

Konfigurations der numerischen Integration

-c, --config: Gibt die zu verwendende Konfigurationsdatei an. Fehlt dieser Parameter, dann werden Standardwerte benutzt. (Die verwendete Konfiguration zeigt das Programm auf STDERR an.)

-C, --read_config_from_input: Statt einer Konfigurationsdatei werden die Einstellungen direkt aus der Eingabedatei gelesen.

Erwartet wird ein Block in demselben Format wie die Konfigurationsdatei, eingeleitet mit der Zeile „config“ und abgeschlossen mit einer Leerzeile. (Siehe Beispiele!!!)

--save_config: Erzeugt eine Konfigurationsdatei mit Standardwerten. Diese Datei kann als Ausgangspunkt für eine eigene Konfigurationsdatei dienen. Details zu den verschiedenen Einstellungen werden als Kommentar am Kopf der Datei ausgegeben.

Beeinflussung des Eingabeformats

-P, --fixed_outputpoints: *Es wird nur einmal eine Liste von Ausgabepunkten eingelesen.* Diese werden dann für alle folgenden Strukturen verwendet. Ohne diesen Parameter müssen zu jeder Struktur auch die gewünschten Ausgabepunkte angegeben werden.

-L, --fixed_loads: *Es wird nur einmal eine Liste von Lasten eingelesen.* Diese werden dann für alle folgenden Strukturen verwendet. Ohne diesen Parameter müssen zu jeder Struktur auch die darauf einwirkenden Lasten angegeben werden.

Der Aufruf ohne Parameter bewirkt, dass die Eingabedaten von STDIN gelesen und die Ergebnisse auf STDOUT ausgegeben werden. (Zur Verwendung mit Pipes und Umleitungsoperatoren.) Hat man das Programm versehentlich in diesem Modus gestartet, kann man es durch mehrfache Eingabe von EOF (F6-TASTE, ENTER) oder mit STRG+C beenden.

4 Exit-Codes

Exit-Code = 0: Fehlerfreie Ausführung. **Exit-Code ungleich 0:** Ungültige Parameter oder Fehler beim Lesen/Schreiben (Details werden auf STDERR ausgegeben.) **Anmerkung:** Exit-Code 0 heißt nicht, dass alle Berechnungen fehlerfrei sind. Solche Fehler werden in den ausgegebenen Ergebnissen vermerkt.

5 Ausgabeformate

Das Ausgabeformat wird mit dem Parameter `--format NAME` ausgewählt. Mögliche Werte: `NAME = csv | integrands`.

Neben einem Ausgabeformat für Debugging-Zwecke (s.u.) ist z.Zt. nur die Ausgabe im CSV-Format vorgesehen. Die zum alten LEAP-Reportformat kompatible Ausgabe wurde entfernt.

CSV-Ausgabe

Die CSV-Ausgabe ist die Voreinstellung. Felder werden mit TAB getrennt und Zeichenketten werden in `"` eingefasst. Vor dem ersten Datensatz wird eine einzelne Kopfzeile mit den Feldnamen ausgegeben. Zu jeder Struktur wird je Ausgabepunkt ein Datensatz ausgegeben. Ein Datensatz enthält jeweils die überlagerten Ergebnisse aller Lasten.

Felder

StructId

Eindeutiger Bezeichner der Struktur.

OutPoint

Eindeutiger Bezeichner des Ausgabepunkts.

x, y, z

Koordinaten des Ausgabepunkts.

Layer

Eindeutiger Bezeichner des Layers.

dispX, dispY, dispZ

Ergebnisse (Überlagerung aller beteiligten Lasten).

**stressXX, stressYY, stressZZ, stressXY, stressYZ, stressZX,
stress1, stress2, stress3**

" " " " "

tau1, tau2, tau3

" " " " "

**strainXX, strainYY, strainZZ, strainXY, strainYZ, strainZX,
strain1, strain2, strain3**

" " " " "

gamma1, gamma2, gamma3

" " " " "

relErr

Maximaler relativer Fehler aller Komponenten.

errCode

Maximaler Fehlercode aller Komponenten.

Hauptspannungen

Die Felder *stress1*, *stress2*, *stress3* sind die drei (nicht notwendig verschiedenen) Eigenwerte der symmetrischen Matrix

$$\begin{pmatrix} stressXX & stressXY & stressZX \\ stressXY & stressYY & stressYZ \\ stressZX & stressYZ & stressZZ \end{pmatrix}.$$

Sie werden mit der LAPACK-Funktion DSYEVR [5] berechnet und in aufsteigender Reihenfolge sortiert. In dieser Reihenfolge werden sie auch zur Berechnung von *tau1...3*, *strain1...3* und *gamma1...3* verwendet.

Tests mit der Eigenwert-Funktion des JAMA-Pakets [4] lieferten identische Ergebnisse. Die Implementierung von Symplectic in LEAP [3] nimmt keine Sortierung vor. Sie basiert auf [2, S. 91–93] und kann laut [1] in gewissen Fällen numerisch instabil sein.

Interpretation des relativen Fehlers (Feld „relErr“)

Das Feld *relErr* gibt Auskunft über die Güte der Berechnungen. Ein Wert von $1.0e-5$ bedeutet beispielsweise, dass die Summe der bis zum Erreichen des Abbruchkriteriums berechneten Integrale ca. 5 gültige Stellen besitzt.

Achtung: Der Betrag des Restglieds (Integral vom letzten verwendeten x bis unendlich) wird nicht geschätzt.

Falls *relErr* kleiner (= besser) als der in der Konfiguration für das Konvergenzkriterium angegebene Wert *eps_convergence* ist, wird *eps_convergence* zurückgegeben.

Die Integration der verschiedenen Komponenten erfolgt stückweise zwischen den Nullstellen der Besselfunktion $J_1(x)$ bis ein Konvergenzkriterium (siehe Konfigurationsdatei) erreicht ist.

Zur Integration der einzelnen Intervalle wird die QUADPACK-Routine DQAGE [6] verwendet. Es handelt sich um eine adaptive Integrationsroutine, die versucht, eine vorgegebene Genauigkeit zu erreichen (siehe Konfigurationsdatei). Gelingt das nicht, wird ein Fehlercode ausgegeben. Die Integrationsroutine liefert außerdem jeweils eine Schätzung des absoluten Fehlers *absErr*. Die absoluten Fehler aller Teil-Integrale werden zu *sumAbsErr* summiert und daraus der Wert des relativen Fehlers berechnet:

$$relErr1 := sumAbsErr / |sum|.$$

Diese Rechnung wird für alle Komponenten durchgeführt, *relErr1* erhält am Ende den maximalen relativen Fehler aller Komponenten.

Zur Vermeidung von Auslöschungseffekten wird außerdem der während der Berechnungen vorkommende „Dynamikumfang“ beobachtet, indem der maximale Absolutbetrag I_maxAbs aller Teilintervalle betrachtet und mit der Endsumme *sum* verglichen wird, wieder jeweils einzeln für alle Komponenten:

$$relErr2 := epsrel * I_maxAbs / |sum| ,$$

wobei *epsrel* der in der Konfigurationsdatei vorgegebene relative Fehler pro Integrationsintervall ist.

Als Feld *relErr* wird abschließend das Maximum aus *relErr1*, *relErr2* und *eps_convergence* zurückgegeben.

Beispiel: Ist $epsrel = 1.0e-8$, dann hat jedes Teil-Integral mindestens

8 gültige Stellen. Ist nun $I_{maxAbs} = 1.23e+2$ und $|sum| = 2.34e-10$, also das Endergebnis um 12 Größenordnungen geringer als das größte Teil-Integral, dann ist $relErr2 > 1$, und damit hat das Ergebnis höchstwahrscheinlich keine korrekten Stellen.

Achtung: Ein Fehlercode (*errCode*, siehe unten) wird niemals auf Grundlage des Wertes von *relErr* ausgegeben. Fehlercodes *errCode* ungleich 0 werden dann ausgegeben, wenn die Integration eines Teilintervalls nicht möglich war oder wenn das Konvergenzkriterium nicht erreicht werden konnte.

Bedeutung der Fehlercodes (Feld „errCode“)

Wenn die in der Konfigurationsdatei vorgegebenen Genauigkeiten oder das Abbruchkriterium nicht erreicht werden konnten, gibt die Integrationsroutine einen Fehlercode ungleich 0 zurück. Das Feld „errCode“ enthält das Maximum aller bei der Integration der verschiedenen Komponenten aufgetretenen Fehlercodes. Falls in der Ausgabedatei im Feld „errCode“ Werte ungleich 0 ausgegeben wurden, kann man versuchen, die Konfiguration oder die Eingabe anzupassen und die Berechnung mit den neuen Vorgaben zu wiederholen, bis alle Fehlercodes = 0 geworden sind.

Mögliche Fehlercodes

| | |
|--|--|
| 0: | Integration erfolgreich. |
| >=100: | Abbruch nach Integration des ersten Intervalls $[0, x_0]$, wobei x_0 die erste Nullstelle von J_1 bezeichnet. |
| >=200: | Abbruch während Integration der restlichen Intervalle. |
| 700: | Alle J_1 -Intervalle durchlaufen, ohne Konvergenz zu erreichen. |
| 800: | Maximal mögliches x erreicht ohne Konvergenz. Mögliche Abhilfe: Den Layer mit maximaler Dicke in 2 Layer halber Dicke aufteilen. |
| 900: | Nicht vorhergesehener Fehler! |
| Fehlercodes von DQAGE werden addiert: | |
| ier = 1 | maximum number of subdivisions allowed has been achieved. (Anm.: Ohne die gewünschte Genauigkeit zu erreichen.) |
| = 2 | the occurrence of roundoff error is detected, which prevents the requested tolerance from being achieved. |
| = 3 | extremely bad integrand behaviour occurs at some points of the integration interval. |
| = 6 | Fehlerhafte Aufrufparameter angegeben. |
| Weitere Fehlercodes, die addiert werden: | |
| = 8 | Maximal erlaubte Anzahl Funktionsauswertungen überschritten. |

Mögliche Maßnahmen beim Auftreten von Fehlercodes

| <i>Fehlercode</i> | <i>Erklärung/Abhilfe</i> |
|-----------------------|---|
| 0 | Alles in Ordnung. |
| Endziffer 1 oder 8 | Erlaubte Anzahl von Funktionsauswertungen überschritten. Maßnahme: <i>maxEval</i> erhöhen. |
| 700 | Für diesen Ausgabepunkt kann keine Konvergenz erreicht werden. Möglicherweise liegt er zu nah an der Oberfläche aber nicht innerhalb des ersten Layers. Eine Sonderbehandlung wie in <i>BISAR</i> [7] kann nur für Punkte im Oberflächen-Layer durchgeführt werden. Abhilfe: Eingabe dahingehend prüfen, ob ein unsinnig dünner oberster Layer benutzt wurde. Layer vergrößern, so dass er diesen Punkt mit einschließt. |
| 800 | Keine Konvergenz erreicht, aber noch nicht alle verfügbaren J_1 -Intervalle durchlaufen. Abbruch auf Grund der Überschreitung des verfügbaren Gleitpunktzahlen-Wertebereichs. <i>Maßnahme</i> : Den dicksten Layer in 2 Layer halber Dicke aufteilen und Berechnung wiederholen. |
| 900 | Unvorhergesehener Fehler bei Auswertung eines Integranden. Vermutlich ein Fehler im Programm! |

Funktionswerte-Ausgabe

Vorgesehen für Debugging-Zwecke und zur weiteren Untersuchung der mathematischen Eigenschaften der beteiligten Funktionen.

Mit dem Ausgabeformat-Parameter **--format integrands** wird, anstatt die Integration durchzuführen, eine Wertetabelle der Integranden ausgegeben.

Ausgabeformat: Die Datensätze werden im TAB-getrennten CSV-Format ausgegeben.

Ausgegebene Felder: *StructId*, *LoadId*, *OutPoint*, *Layername*, *A*, *B*, *C*, *D* (die 4 Koeffizienten des Layers), *x* (Integrationsvariable), *Srr*,

S_{tt} , S_{zz} , S_{zr} , u , w . (Stress- und Displacement-Komponenten im zylindrischen Koordinatensystem.)

Reihenfolge der Ausgabe: Die Ausgabe erfolgt in 3 verschachtelten Schleifen über alle Strukturen, alle Lasten und alle Ausgabepunkte. (Weil dabei größere Datenmengen entstehen als bei der normalen Ausgabe, ist es empfehlenswert, sich auf ein einzelnes System mit nur einer Last zu beschränken.)

Die Integrationsvariable x läuft von 0 bis zum Minimum aus verfügbarem Gleitpunktzahlen-Wertebereich und letzter verfügbarer $J_1(x)$ -Nullstelle.

6 Konfiguration

– *Dokumentation ergänzen!* –

maxEval

Gibt die Anzahl der Funktionsauswertungen an, die der Integrationsroutine pro zu integrierender Funktion einer Szene zugebilligt werden. Wurde dieses Kontingent ausgeschöpft, ohne dass Konvergenz erreicht ist, dann wird das bis dahin ermittelte Ergebnis ausgegeben und ein Fehlercode *errCode* ungleich 0.

Wert muss > 0 sein.

epsabs

Wert > 0 : Maximal erlaubter absoluter Fehler je Integrationsintervall. Wenn die Integrationsroutine diesen Wert in einem Intervall nicht erreichen konnte, wird ein Fehlercode *errCode* ungleich 0 ausgegeben.

Wert $= -1$: Kein absolutes Fehlerkriterium verwenden.

~~Es ist empfehlenswert, kein absolutes Kriterium zu verwenden,~~
weil sonst die Ergebnisse von den in den Eingabedaten benutzten Einheiten abhängig sind.

epsrel

Wert $0 < epsrel \leq 1$: Maximal erlaubter relativer Fehler je Integrationsintervall. Fehlercode ungleich 0, wenn dies nicht erreicht werden konnte.

Wert ≥ 1 : Kein relatives Kriterium verwenden. Wenn zugleich $epsabs = -1$ gewählt wird, dann wird die Adaptivität abgeschaltet: Die Integrationsroutine gibt sich mit jedem Ergebnis zufrieden.)

intkey1

Auswahl der Quadraturformel für das erste Teilintervall $[0, J_1(x_0)]$ (s.u.).

intkey2

Auswahl der Quadraturformel für die übrigen Teilintervalle (s.u.).

eps_convergence

*Konvergenzkriterium: Das Kriterium ist erfüllt, wenn
[Dokumentation ergänzen!]*

convergence_count

Gibt die Anzahl der aufeinanderfolgenden Intervalle an, die das Konvergenzkriterium erfüllen müssen, bevor von Konvergenz ausgegangen wird.

Wert ≤ 1 : Das Konvergenzkriterium muss genau einmal erfüllt sein.

Verfügbare Integrationsroutinen

Die beiden Parameter *intkey1* und *intkey2* legen fest, welche Quadraturformeln die adaptiven Integrationsroutine *DQAGE* [6] benutzen soll. Dabei bezieht sich *intkey1* auf das erste Teilintervall $[0, J_1(x_0)]$ von 0 bis zur ersten Nullstelle von $J_1(x)$, das i.d.R. schwieriger zu integrieren ist als die übrigen Teilintervalle. Deren Quadraturformel wird mit *intkey2* bestimmt.

Die Wahl dieser Parameter beeinflusst im wesentlichen die Rechenzeit, weniger die Güte der Berechnungen: DQAGE führt bei Be-

darf eine Bisektion durch, bis die mittels epsrel vorgeschriebene Genauigkeit erreicht ist.

Zur Minimierung der Rechenzeit sind die Parameter so zu wählen, dass einerseits möglichst keine Bisektion nötig wird (Formeln höheren Grades wählen), aber zugleich nicht unnötig viele Funktionsauswertungen durchgeführt werden (Formeln niedrigeren Grades bevorzugen). Die optimale Wahl der Parameter ist von der Szene und den Positionen der Ausgabepunkte abhängig.

| intkey | QUADPACK-Funktion (Gauss-Kronrod-Quadratur) |
|--------|--|
| 1 | DQK15 (15-Punkt-Formel) |
| 2 | DQK21 (21-Punkt-Formel) |
| 3 | DQK31 (31-Punkt-Formel) |
| 4 | DQK41 (41-Punkt-Formel) |
| 5 | DQK51 (51-Punkt-Formel) |
| 6 | DQK61 (61-Punkt-Formel) |

Beispiel für eine Konfigurationsdatei

```
maxEval = 5000 # Maximal erlaubte Anzahl von Funktionsauswertungen.
epsabs = -1.0  # Absolutes Fehlerkriterium nicht verwenden.
epsrel = 1.0e-7 # Gewünschte Genauigkeit 7 Stellen.
intkey1 = 6    # DQK61 für erstes Intervall verwenden.
intkey2 = 2    # DQK21 für restliche Intervalle verwenden.

# Konvergenzkriterium: Zuwachs < 1e-4 vom Durchschnitt:
eps_convergence = 1.0e-4

# Kriterium muss für 2 aufeinanderfolgende Intervalle erfüllt sein:
convergence_count = 2
```

Beispiel für „LEAP-ähnliche“ Einstellungen

```
maxEval = -1
```

```
epsabs = -1  
epsrel = 1  
intkey1 = 1  
intkey2 = 1  
eps_convergence = 1e-2  
convergence_count = 1.
```



7 Format der Eingabedatei

Das Eingabeformat orientiert sich an dem von *LEAP* [3], wurde aber in einigen Punkten modifiziert:

- Die Eingabe erfolgt in der festen Reihenfolge *Ausgabepunkte*, *Lasten*, *Layer*.
- Der erlaubte Wertebereich für *ny* ist nun $[-1, 1]$ statt wie in LEAP $-1 \leq ny < 0.5$.
- Kommentare können mit „#“ eingefügt werden und gelten bis zum Ende der Zeile.

Formale Syntax

```
Szene :=  
    Ausgabepunkte-Block  
    +<NEWLINE>  
    Lasten-Block  
    +<NEWLINE>  
    Struktur-Block  
    (+<NEWLINE> | *<NEWLINE><EOF>)  
  
Ausgabepunkte-Block :=  
    „outpoint“  
    +Ausgabepunkt  
  
Lasten-Block :=  
    „load“
```

```

+Last

Struktur-Block :=
    struct [Struktur-Name]
    +Layer

Ausgabepunkt :=
    „add“ Punkt-Name x y z <NEWLINE>

Last :=
    „add“ Last-Name x y radius Fx Fy Fz <NEWLINE>

Layer :=
    „add“ Layer-Name E ny height flag spring <NEWLINE>

flag := 0 | 1

```

Beispiel einer Eingabedatei

```

outpoint
add PunktA 0 0 0.20          # Ausgabepunkt in 0.20 m Tiefe (Layer L0).
add PunktB 0 0 0.200000001 # Ausgabepunkt in 0.20 m Tiefe (Layer L1).

load
add load 0.0 0.0 0.15  0 0 0.03 # Senkrechte zentrale Last.

struct 0 # Struktur mit der ID 0
add L0 20000 0.35 0.2 0 0.0
add L1 18000 0.35 0.2 0 0.0
add L2 18000 0.35 0.2 0 0.0
add Linf 120 0.50 0.0 0 0.0

```

Anmerkungen zur Notation:

- „+“ heißt, dass das nachfolgende Element mindestens einmal vorkommt, „*“, dass das Element beliebig oft (auch gar nicht) vorkommen kann. In „[]“ eingefasste Elemente sind optional, das Zeichen „|“ bedeutet eine Alternative.
- Mit <NEWLINE> ist ein Zeilenvorschub gemeint; erlaubt ist sowohl ein *Carriage-Return+Linefeed* als auch ein einfaches *Linefeed*. <EOF> bezeichnet das Dateiende.
- Die in Anführungszeichen eingefassten Zeichen werden so, wie sie stehen (ohne die Anführungszeichen selbst),

eingegeben.

- Abweichend von der folgenden Syntax-Definition werden bei Verwendung der Parameter *--fixed_outputpoints* und *--fixed_loads* die jeweiligen Blöcke für Ausgabepunkte oder Lasten nur einmalig (bei der ersten Szene) angegeben. Bei den übrigen Szenen entfällt die Angabe.

Ausgabepunkte auf der Grenzfläche zwischen zwei Layern

Die Unterkante eines Layers und die Oberkante des nachfolgenden Layers haben eine gemeinsame z-Koordinate. Es gilt die folgende Regelung:

Ausgabepunkte mit z-Koordinaten auf der Grenzfläche zählen zum oberen Layer.

Um einen Ausgabepunkt zu definieren, der direkt an der Oberkante eines Layers liegt, muss ein winziger Betrag zur z-Koordinate addiert werden. Im Beispiel oben liegt der Ausgabepunkt *PunktA* im Layer *L0*, und *PunktB* liegt im Layer *L1*.



Falls der addierte Betrag zu klein gewählt wird, geht er beim Lesen der Eingabedatei durch Rundung verloren. Die für die Eingabe verwendeten *double-Variablen* können etwa 14 Dezimalstellen wiedergeben. Ob der Ausgabepunkt tatsächlich dem beabsichtigten Layer zugeordnet wurde, kann am Feld *Layer* in der CSV-Ausgabe abgelesen werden.

8 Multithreading

Der Aufrufparameter `--threads N` bewirkt, dass N parallele Threads ausgeführt werden. Fehlt der Parameter, dann wird $N = 1$ verwendet. Bei $N = 0$ wird N automatisch auf die Anzahl der verfügbaren CPUs eingestellt.

Die Parallelisierung findet auf der Ebene von Szenen statt, d.h., die Verwendung von N Threads lohnt sich nur, wenn auch mindestens N zu berechnende Szenen vorhanden sind.

Es schadet nicht, die Anzahl höher als die der verfügbaren CPUs zu setzen: Dann findet zwar keine Leistungssteigerung statt, aber auch kein Performance-Einbruch. Jedoch wird entsprechend mehr Speicher benötigt.

Die Reihenfolge der Ergebnisse ist bei $N > 1$ Threads nicht determiniert: Die einzelnen Szenen werden ausgegeben, wenn sie fertig berechnet sind. Ihre Reihenfolge ist nicht vorhersagbar. Die Ausgabepunkte der jeweiligen Szene werden dagegen stets der Reihenfolge ihrer Definition in der Eingabedatei ausgegeben.

9 Grenzen und Einschränkungen des Programms

- Anders als in *BISAR* [7] und *LEAP* [3] können zur Zeit *keine Lasten mit horizontalen Kraftanteilen* verwendet werden. Diese Funktion wurde wegen sich abzeichnender numerischer Probleme vorerst deaktiviert. Die Eingabe derartiger Lasten führt zu einer Fehlermeldung.
- Dagegen findet seit der letzten Version nun wie in *BISAR* eine

gesonderte Behandlung des Falls oberflächennaher Ausgabe-
punkte mittels spezieller Berechnungsmethode statt. (Die in
LEAP verwendete Berechnungsmethode versagt bei Ausgabe-
punkten, deren z-Koordinate nahe 0 liegt: Das Konvergenz-
kriterium wird dann nicht erreicht, oder es werden unsinnige
Ergebnisse geliefert.) Diese Sonderbehandlung ist, ebenfalls
wie in BISAR, jedoch auf den obersten Layer beschränkt.

Literatur

- [1] W. M. Scherzinger, C.R. Dohrmann. A robust algorithm for finding the eigenvalues and eigenvectors of 3x3 symmetric matrices. *Comput. Methods Appl. Mech. Engrg.* 197 (2008) 4007–4015.
- [2] L. E. Malvern. *Introduction to the Mechanics of a Continuous Medium*. Englewood Cliffs, NJ: Prentice-Hall, 1969.
- [3] Symplectic Engineering Corporation. LEAP 1.0. Berkeley, California. [Java-Quellcode, 1997–99]
- [4] National Institute of Standards and Technology (NIST). "JAMA: A Java Matrix Package". <http://math.nist.gov/javanumerics/jama/>.
- [5] Netlib. "LAPACK – Linear Algebra PACKage". <http://www.netlib.org/lapack/>.
- [6] R. Piessens, E. de Doncker-Kapenger, C. Ueberhuber, D. Kahaner. *QUADPACK, A Subroutine Package for Automatic Integration*. Springer: 1983. FORTRAN77-Quellcode von <http://www.netlib.org/quadpack/>.
- [7] De Jong, D. L., M. G. F. Peutz und A. R. Korswagen. *Computer Program BISAR: Layered systems under normal and tangential loads*. AMSR.0006.73. External Report. (January 1973) Shell Research B. V. Amsterdam: Koninklijke/Shell-Laboratorium, 1979.