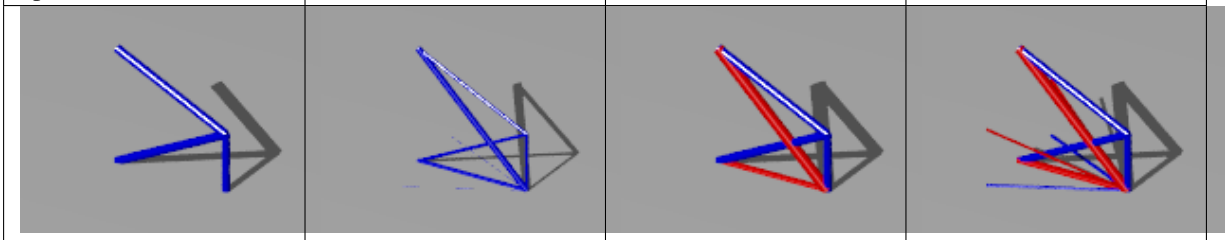
ttrr Documentation

Release 2013-11-09 (alpha)

Daniel Mohr

November 09, 2013

CONTENTS

optimal	robust	redundant	robust + redundant
			
			see *Example 3*

ttrr is an acronym for Truss Topology Robust Redundant.

ttrr is a software package to do topology optimization of a truss with respect to robustness and redundancy.

This documentation is available as:

- html http://ttrr.cadae.de/ (incl. downloads)
- pdf: `ttrr.pdf`
- epub: `ttrr.epub`

Contents:

CHAPTER

ONE

DOWNLOAD

The available versions are:

- `ttrr-latest.tar.gz` (alpha)
  md5sum:
- `ttrr-2012-06-26.tar.gz` (alpha)
  md5sum: f4323db1c107379a664c65fcdfa265fb
- `ttrr-2012-05-05.tar.gz` (alpha)
  md5sum: 421f4e94be1cb74f5068224cf83f8af2
- `ttrr-2012-03-16.tar.gz` (alpha)
  md5sum: cd26fb4d145d24f87b4b061993e08162
- `ttrr-2012-03-10.tar.gz` (alpha)
  md5sum: 579975a0f99ce6b3efb6d69f95d03c7c
- ttrr-2012-03-09.tar.gz (alpha)
  md5sum: ed4528c399c0e6bef617c4045ab00b53
- ttrr-2012-02-23.tar.gz (alpha)
  md5sum: 6f5a4f57ba761b923ea6828aeee53057
- ttrr-2012-02-22.tar.gz (pre-alpha)
  md5sum: a007c94ae1e72236d49b6b30d16e266a

CHAPTER

TWO

TTRR – TRUSSTOPOLOGYROBUS-TREDUNDANT

platform Unix

synopsis ttrr is a software package to do topology optimization of a truss with respect to robustness and redundancy.

TrussTopologyRobustRedundant

Contents

2.1 description

ttrr is a software package to do topology optimization of a truss with respect to robustness and redundancy — see literatur.

There are 2 parts of ttrr:

- **library:** You can use `ttrr` as a library to import to your own python-code. To calculate trusses `ttrr` needs **ttrr_calculate** resp. ttrr_calculate.cc
- **command line tool:** you can use `ttrr_tools.py` as a command line tool. for detail:

```
ttrr_tools.py -h
```

functions to include from ttrr resp. ttrr.py: (in alphabetical order)

- `compute_disturbed_forces()`
- `nodenumberfromcoordinates()`
- `barsnodesongrid2d()`
- `barsnodesongrid3d()`
- `ttrr()`
- `ttrr_redundant()`
- `ttrr_robust()`

2.2 typical parameters

dim [integer] dimension 2 or 3 of the design space

nodes [numpy.array] is a matrix to represent all nodal points with:

- nodes(k,:) are the nodal points.
- nodes(:,1) are the x-coordinates of the nodal points.
- nodes(:,2) are the y-coordinates of the nodal points.
- nodes(:,3) are the z-coordinates of the nodal points. (mostly optional)

bars [numpy.array] is a vector to represent all trusses:

- bars(s,:) are numbers of the trusses.
- bars(:,1) is the number of the first nodal point.
- bars(:,2) is the number of the second nodal point.

mounting [numpy.array] are the numbers of the nodal points as for the coordinate direction which are fixed.

loads [scipy.sparse.lil_matrix] are the load forces.

loads(:,k) are the forces for the k-th load case

smax [float] is the maximum allowed cross section of a truss

opt [integer] choose the optimization routine: (optional)

- 0 glp_simplex (default, iff redundancy == 0)
- 1 glp_interior (only for redundancy == 0)
- 2-5 glp_mip (for redundancy != 0)
  - 2 do not search greedy-solution (default, for redundancy != 0)
  - 3 search greedy-solution and go on
  - 4 search greedy-solution, write greedy solution and go on
  - 5 search greedy-solution and stop

2.3 functions

ttrr.**compute_disturbed_forces**(*loads*, *dim*, *p*, *art=0*)

forces_disturbed = compute_disturbed_forces(loads,dim,p,art=0)

calculates perturbed forces from loads and returns them forces_disturbed. For nl load cases in loads you get in forces_disturbed (2^dim)*nl load cases.

Parameters:

dim [integer] dimension 2 or 3 of the design space

p [float] percentage of magnitude of the unperturbed force is used for the perturbation. p*norm(loads[:,k]) is the magnitude of the perturbation of the load case k.

art [integer] chooses the kind of the perturbation: (optional)

art == 0 generates perturbations in every direction of the original forces and scales them. The maximal magnitude of the forces is the same. (default)

art == 1 generates perturbations in every direction of the original forces. The maximal magnitude of the forces changes.

art [integer] waehlt die Art die gestoerrten Kraefte zu erstellen (optional)

art == 0 erzeuge Stoerrung um p*norm(kraft) in jede Richtung. skaliere die Kraefte so, dass die maximale Kraftgroesse gleich bleibt (default)

art == 1 erzeuge Stoerrung um p*norm(kraft) in jede Richtung. Achtung: maximale Kraftgroesse aendert sich

Author: Daniel Mohr.

Date: 2012-01-23, 2012-01-28, 2012-02-27 (last change)

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

Copyright (C) 2011, 2012 Daniel Mohr This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, see http://www.gnu.org/licenses/

`ttrr.`**`nodenumberfromcoordinates`**(*nodes*, *x*, *y*, *z=0*, *eps=1.4901161193847656e-08*)

kk = nodenumberfromcoordinates(nodes,x,y,z=0,eps=_epsilon) calculates the number of a node from the coordinates.

nodes is a matrix to represent all nodal points with nodes(k,:) are the nodal points nodes(:,1) are the x-coordinates of the nodal points nodes(:,2) are the y-coordinates of the nodal points nodes(:,3) are the z-coordinates of the nodal points (optional)

x,y,z gives the coordinate of the wanted node. If the dimension of the design space (it is get from the dimension of nodes) is 2 z is ignored. z is optional.

eps is a bound for the possible difference between the coordinate of the wanted node and given coordinate.

Author: Daniel Mohr Date: 2012-01-23, 2012-01-28 (last change) License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007

Copyright (C) 2012 Daniel Mohr This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, see http://www.gnu.org/licenses/

`ttrr.`**`barsnodesongrid2d`**(*gx0*, *gx1*, *gy0*, *gy1*, *nelx*, *nely*, *mounting=0*)

[nodes,bars]=barsnodesongrid2d(gx0,gx1,gy0,gy1,nelx,nely,mounting=0)

creates all bars and nodes which make sense. gx0, gx1, gy0 and gy1 describes the edges (gx0,gy0), (gx0,gy1), (gx1,gy0) and (gx1,gy1) of the rectangular design space. In this rectangle it will be nelx*nely nodes calculated. With the assumption of an equidistant grid all meaningful bars are calculated. Bars which can be represent of others are ignored. For example a bar from (0,0) to (0,1) is ignored, if bars from (0,0) to (0,0.5) and from (0,0.5) to (0,1) exist.

Parameters:

gx0 [float] lower bound of the x-cooredinates to describe the rectangle

gx1 [float] upper bound of the x-cooredinates to describe the rectangle

gy0 [float] lower bound of the y-cooredinates to describe the rectangle

gy0 [float] upper bound of the y-cooredinates to describe the rectangle

nelx [integer] is the number of nodes in x-direction

nely [integer] is the number of nodes in y-direction

mounting [numpy.array] are the numbers of the nodal points as for the coordinate direction which are fixed. (optional) If denoted bars fixed on both ends will be ignored.

Returns:

nodes [numpy.array] is a matrix to represent all nodal points with:

- nodes(k,:) are the nodal points.
- nodes(:,1) are the x-coordinates of the nodal points.
- nodes(:,2) are the y-coordinates of the nodal points.
- nodes(:,3) are the z-coordinates of the nodal points. (mostly optional)

bars [numpy.array] is a vector to represent all trusses:

- bars(s,:) are numbers of the trusses.
- bars(:,1) is the number of the first nodal point.
- bars(:,2) is the number of the second nodal point.

copyright + license

Author: Daniel Mohr.

Date: 2010-09-09, 2011-06-24, 2012-01-24 (last change)

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

Copyright (C) 2009, 2010, 2011, 2012 Daniel Mohr This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, see http://www.gnu.org/licenses/

`ttrr.`**`barsnodesongrid3d`**(*gx0*, *gx1*, *gy0*, *gy1*, *gz0*, *gz1*, *nelx*, *nely*, *nelz*, *mounting=0*, *areawithoutnode=0*, *mountingarea=0*, *maxstablaenge=-1*)

[nodes,bars]=barsnodesongrid3d(gx0,gx1,gy0,gy1,gz0,gz1,nelx, nely,nelz,mounting=0,areawithoutnode=0,maxstablaenge=-1) [nodes,bars,mounting]=barsnodesongrid3d(gx0,gx1,gy0,gy1,gz0,gz1,nelx,

nely,nelz,areawithoutnode=0,mountingarea=0, maxstablaenge=-1)

creates all bars and nodes which make sense. If mountingarea is denoted the fixed nodes are also calculated and returned. gx0, gx1, gy0, gy1, gz0 and gz1 describes the edges (gx0,gy0,gz0), (gx0,gy0,gz1), (gx0,gy1,gz0), (gx0,gy1,gz1) (gx1,gy0,gz0), (gx1,gy0,gz1), (gx1,gy1,gz0) and (gx1,gy1,gz1) of the cuboid-shaped design space. In this cuboid it will be nelx*nely*nelz nodes calculated. With the assumption of an equidistant grid all meaningful bars are calculated. Bars which can be represent of others are ignored. For example a bar from (0,0,0) to (0,0,2) is ignored, if bars from (0,0,0) to (0,0,1) and from (0,0,1) to (0,0,2) exist.

Parameters:

gx0 [float] lower bound of the x-cooredinates to describe the rectangle

gx1 [float] upper bound of the x-cooredinates to describe the rectangle

gy0 [float] lower bound of the y-cooredinates to describe the rectangle

gy0 [float] upper bound of the y-cooredinates to describe the rectangle

gz0 [float] lower bound of the z-cooredinates to describe the rectangle

gz0 [float] upper bound of the z-cooredinates to describe the rectangle

nelx [integer] is the number of nodes in x-direction

nely [integer] is the number of nodes in y-direction

nelz [integer] is the number of nodes in z-direction

mounting [numpy.array] are the numbers of the nodal points as for the coordinate direction which are fixed. (optional) If denoted bars fixed on both ends will be ignored.

areawithoutnode [numpy.array] is a matrix describing areas without nodes (optional) In every row are 6 values ax, bx, ay, by, az, bz. Nodes with coordinate (kx,ky,kz) and (ax<=kx<=bx and ay<=ky<=by and az<=kz<=bz) will be ignored for the bars. Theses nodes are in nodes. But no bar will be connect to these nodes.

mountingarea [numpy.array] is a matrix describing areas where the nodes are fixed. (optional) In every row are 6 values ax, bx, ay, by, az, bz. Nodes with coordinate (kx,ky,kz) and (ax<=kx<=bx and ay<=ky<=by and az<=kz<=bz) are considered as fixed nodes. If mountingarea is denoted also mounting will be returned. The paarameter mounting will be ignored. But the affected bars will be ignored.

maxstablaenge [float] Bars which are longer than maxstablaenge will be ignored.

Returns:

nodes [numpy.array] is a matrix to represent all nodal points with:

- nodes(k,:) are the nodal points.
- nodes(:,1) are the x-coordinates of the nodal points.
- nodes(:,2) are the y-coordinates of the nodal points.
- nodes(:,3) are the z-coordinates of the nodal points.

bars [numpy.array] is a vector to represent all trusses:

- bars(s,:) are numbers of the trusses.
- bars(:,1) is the number of the first nodal point.
- bars(:,2) is the number of the second nodal point.

mounting [numpy.array] are the numbers of the nodal points as for the coordinate direction which are fixed. (optional) It will will be returned only if mountingarea is denoted.

copyright + license

Author: Daniel Mohr.

Date: 2011-06-24, 2012-01-24, 2012-01-25 (last change)

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

Copyright (C) 2011, 2012 Daniel Mohr This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, see http://www.gnu.org/licenses/

`ttrr.`**`ttrr`**(*dim*, *nodes*, *bars*, *mounting*, *loads*, *smax*, *name='test'*, *sigmamin=-100000000.0*, *sigmamax=100000000.0*, *density=2700.0*, *redundancy=0*, *opt=0*, *undisturbed_forces=0*, *sort=0*, *zlb=0*, *zub=0*, *scale_sigma=-1.0*, *scale_forces=-1.0*)

[s,w,lengths] = ttrr(dim,nodes,bars,mounting,loads,smax,name='test', sigmamin=-1e8,sigmamax=1e8,density=2.7e3,redundancy=0,

opt=0,undisturbed_forces=0,sort=0,zlb=0,zub=0, scale_sigma=-1,scale_forces=-1)

Parameters:

dim [integer] dimension 2 or 3 of the design space

nodes [numpy.array] is a matrix to represent all nodal points with:

- nodes(k,:) are the nodal points.
- nodes(:,1) are the x-coordinates of the nodal points.
- nodes(:,2) are the y-coordinates of the nodal points.
- **nodes(:,3) are the z-coordinates of the nodal points.** (optional for dim = 2)

bars [numpy.array] is a vector to represent all trusses:

- bars(s,:) are numbers of the trusses.
- bars(:,1) is the number of the first nodal point.
- bars(:,2) is the number of the second nodal point.

mounting [numpy.array] are the numbers of the nodal points as for the coordinate direction which are fixed.

loads [scipy.sparse.lil_matrix] are the load forces.

loads(:,k) are the forces for the k-th load case

smax [float] is the maximal allowed cross section of a bar. If you do not need it, set it to a huge value — e. g. 1.

name [string] is a name for the problem. The result is saved in name.tar.bz2 in the current working directory.

sigmamin [float] lower limit for the allowed stress, e.g. compressive strength (e.g. -1e8 Pa for aluminium or -4e8 Pa for some steel)

sigmamax [float] upper limit for the allowed stress, e.g. tensile strength (e.g. 1e8 Pa for aluminium or 4e8 Pa for some steel)

density [float] density (e.g. 2.7e3 kg/m^3 for aluminium or 7e3 kg/m^3 for some steel)

redundancy [integer] describes the redundancy. If redundancy != 0 this function is not the right one. `ttrr_redundant()` will be called.

opt [integer] choose the optimization routine: (optional)

- 0 glp_simplex (default, iff redundancy == 0)
- 1 glp_interior (only for redundancy == 0)
- **2-6 glp_mip (only for redundancy != 0, therefore not in this** function) see `ttrr_redundant()`

undisturbed_forces [numpy.array] is not used. It will be saved to the problem data.

sort [integer] describes a feature: (optional)

- 0 not used (default)
- **1 additional constraints will be added to the optimization** problem. This is only useful for redundancy != 0. see `ttrr_redundant()`

zlb [float] is a lower bound for the objective function. As we are minimizing the result must not the best solution. Only values greater zero are considered. (optional)

zub [float] is a upper bound for the objective function. As we are minimizing the optimization problem may be not solvable. Only values greater zero are considered. (optional)

scale_sigma [float]

scales sigmamin and sigmamax for the calculation. (optional)

- < 0.0 no scaling will be done (default)
- = 0.0 autoscaling will be done
- > 0.0 scaling will be done with this value

scale_forces [float]

scales loads for the calculation. (optional)

- < 0.0 no scaling will be done (default)
- = 0.0 autoscaling will be done
- > 0.0 scaling will be done with this value

copyright + license

Author: Daniel Mohr.

Date: 2012-02-19 (last change)

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

Copyright (C) 2012 Daniel Mohr This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, see http://www.gnu.org/licenses/

ttrr.**ttrr_redundant**(*dim*, *nodes*, *bars*, *mounting*, *loads*, *smax*, *name='test'*, *sigmamin=-100000000.0*, *sigmamax=100000000.0*, *density=2700.0*, *redundancy=0*, *opt=2*, *undisturbed_forces=0*, *sort=0*, *zlb=0*, *zub=0*, *scale_sigma=-1.0*, *scale_forces=-1.0*)

[s,w,exitflag,zeit,lengths] = ttrr_redundant(dim,nodes,bars,mounting, loads,smax,name='test',sigmamin=-1e8,

sigmamax=1e8,density=2.7e3,redundancy=0, opt=2,undisturbed_forces=0,sort=0, zlb=0,zub=0, scale_sigma=-1,scale_forces=-1)

Parameters:

dim [integer] dimension 2 or 3 of the design space

nodes [numpy.array] is a matrix to represent all nodal points with:

- nodes(k,:) are the nodal points.
- nodes(:,1) are the x-coordinates of the nodal points.
- nodes(:,2) are the y-coordinates of the nodal points.
- **nodes(:,3) are the z-coordinates of the nodal points.** (optional for dim = 2)

bars [numpy.array] is a vector to represent all trusses:

- bars(s,:) are numbers of the trusses.
- bars(:,1) is the number of the first nodal point.
- bars(:,2) is the number of the second nodal point.

mounting [numpy.array] are the numbers of the nodal points as for the coordinate direction which are fixed.

loads [scipy.sparse.lil_matrix] are the load forces.

loads(:,k) are the forces for the k-th load case

smax [float] is the maximal allowed cross section of a bar. If you do not need it, set it to a huge value — e. g. 1. It is nessecary for the interaction of the integer values and the cross section values (for redundancy != 0).

name [string] is a name for the problem. The result is saved in name.tar.bz2 in the current working directory.

sigmamin [float] lower limit for the allowed stress, e.g. compressive strength (e.g. -1e8 Pa for aluminium or -4e8 Pa for some steel)

sigmamax [float] upper limit for the allowed stress, e.g. tensile strength (e.g. 1e8 Pa for aluminium or 4e8 Pa for some steel)

density [float] density (e.g. 2.7e3 kg/m^3 for aluminium or 7e3 kg/m^3 for some steel)

redundancy [integer] describes the redundancy. If redundancy == 0 this function is not the right one. `ttrr()` will be called. A value greater 0 describes a 1/redundancy redundancy — therefore redundancy >= 2 makes sense. A value lower 0 descirbes a 1-1/redundancy redundancy — therefore redundancy <= 2 makes sense. Obviosly 1/2 = 1-1/2 are the same redundancy. But the problem and the calculation are different. If there is no error the results are equal.

opt [integer] choose the optimization routine: (optional)

- 0 glp_simplex (only for redundancy == 0)
- 1 glp_interior (only for redundancy == 0)
- 2-6 glp_mip (for redundancy != 0)
  - 2 do not search greedy-solution (default)
  - 3 search greedy-solution and go on
  - 4 search greedy-solution, write greedy solution and go on
  - 5 search greedy-solution and stop
  - **6 search in parallel with 2 and 3 and stop** after solution is found with one method

undisturbed_forces [numpy.array] is not used. It will be saved to the problem data.

sort [integer] describes a feature: (optional)

- 0 not used (default)
- **1 additional constraints will be added to the optimization** problem. This is only useful for redundancy != 0. If s_i describes a vector of the cross section of the i-th part structure the constraints are l^T s_i <= l^T s_{i+1} for adequate i with l the vector of the lengths of all bars.

zlb [float] is a lower bound for the objective function. As we are minimizing the result must not the best solution. Only values greater zero are considered.

zub [float] is a upper bound for the objective function. As we are minimizing the optimization problem may be not solvable. Only values greater zero are considered.

scale_sigma [float]

scales sigmamin and sigmamax for the calculation. (optional)

- < 0.0 no scaling will be done (default)
- = 0.0 autoscaling will be done
- > 0.0 scaling will be done with this value

scale_forces [float]

scales loads for the calculation. (optional)

- < 0.0 no scaling will be done (default)
- = 0.0 autoscaling will be done
- > 0.0 scaling will be done with this value

copyright + license

Author: Daniel Mohr.

Date: 2013-08-30 (last change)

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

`ttrr.`**`ttrr_robust`**(*dim*, *nodes*, *bars*, *mounting*, *loads*, *smax*, *name='test'*, *sigmamin=-100000000.0*, *sigmamax=100000000.0*, *density=2700.0*, *redundancy=0*, *opt=0*, *undisturbed_forces=0*, *p=0.1*, *art=0*, *sort=0*, *zlb=0*, *zub=0*, *scale_sigma=-1.0*, *scale_forces=-1.0*)

[s,w,exitflag,zeit,lengths] = ttrr_robust(dim,nodes,bars,mounting, loads,smax,name='test',sigmamin=-1e8, sigmamax=1e8,density=2.7e3,redundancy=0, opt=0,undisturbed_forces=0,p=0.1,art=0, sort=0,zlb=0,zub=0, scale_sigma=-1,scale_forces=-1)

Parameters:

dim [integer] dimension 2 or 3 of the design space

nodes [numpy.array] is a matrix to represent all nodal points with:

- nodes(k,:) are the nodal points.
- nodes(:,1) are the x-coordinates of the nodal points.
- nodes(:,2) are the y-coordinates of the nodal points.
- **nodes(:,3) are the z-coordinates of the nodal points.** (optional for dim = 2)

bars [numpy.array] is a vector to represent all trusses:

- bars(s,:) are numbers of the trusses.
- bars(:,1) is the number of the first nodal point.
- bars(:,2) is the number of the second nodal point.

mounting [numpy.array] are the numbers of the nodal points as for the coordinate direction which are fixed.

loads [scipy.sparse.lil_matrix] are the load forces.

loads(:,k) are the forces for the k-th load case

smax [float] is the maximal allowed cross section of a bar. If you do not need it, set it to a huge value — e. g. 1. It is nessecary for the interaction of the integer values and the cross section values (for redundancy != 0).

name [string] is a name for the problem. The result is saved in name.tar.bz2 in the current working directory.

sigmamin [float] lower limit for the allowed stress, e.g. compressive strength (e.g. -1e8 Pa for aluminium or -4e8 Pa for some steel)

sigmamax [float] upper limit for the allowed stress, e.g. tensile strength (e.g. 1e8 Pa for aluminium or 4e8 Pa for some steel)

density [float] density (e.g. 2.7e3 kg/m^3 for aluminium or 7e3 kg/m^3 for some steel)

redundancy [integer] describes the redundancy. After evaluating the special parameter the other functions are called. If redundancy == 0 the function `ttrr()` will be called and if redundancy != 0 the function `ttrr_redundant()` will be called.

opt [integer] choose the optimization routine: (optional)

- 0 glp_simplex (for redundancy == 0) (default)
- 1 glp_interior (for redundancy == 0)
- 2-6 glp_mip (for redundancy != 0)
  - 2 do not search greedy-solution
  - 3 search greedy-solution and go on
  - 4 search greedy-solution, write greedy solution and go on
  - 5 search greedy-solution and stop
  - **6 search in parallel with 2 and 3 and stop** after solution is found with one method

undisturbed_forces [numpy.array] If it is a scalar the perturbed forces are calculated with `compute_disturbed_forces()`. Otherwise loads are assumed to be the perturbed forces. The content of undisturbed_forces is not used. It will be saved to the problem data.

p [float] is a parameter for the calculation of perturbed forces. (optional) see `compute_disturbed_forces()`

art [integer] chooses the kind of the perturbation: (optional) see `compute_disturbed_forces()`

sort [integer] describes a feature: (optional)

- 0 not used (default)
- **1 additional constraints will be added to the optimization** problem. see `ttrr_redundant()`

zlb [float] is a lower bound for the objective function. As we are minimizing the result must not the best solution. Only values greater zero are considered.

zub [float] is a upper bound for the objective function. As we are minimizing the optimization problem may be not solvable. Only values greater zero are considered.

scale_sigma [float]

scales sigmamin and sigmamax for the calculation. (optional)

- < 0.0 no scaling will be done (default)
- = 0.0 autoscaling will be done
- > 0.0 scaling will be done with this value

scale_forces [float]

scales loads for the calculation. (optional)

- < 0.0 no scaling will be done (default)
- = 0.0 autoscaling will be done
- > 0.0 scaling will be done with this value

copyright + license

Author: Daniel Mohr.

Date: 2012-02-19 (last change)

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

2.4 dependencies

Dependencies for ttrr.py and ttrr_tools.py:

- Python http://www.python.org/
- NumPy http://numpy.scipy.org/ http://pypi.python.org/pypi/numpy
- SciPy http://www.scipy.org/ http://pypi.python.org/pypi/scipy
  - and also for NumPy/SciPy: BLAS and LAPACK libraries

Dependencies for ttrr and ttrr.cc, respectively:

- GLPK http://www.gnu.org/software/glpk/

Runtime dependencies for ttrr_tools.py to get graphical output (picture, movie, animation) you need some of the following programs:

- gnuplot http://www.gnuplot.info/
- POV-Ray http://www.povray.org/
- png2yuv from MJPEG http://mjpeg.sourceforge.net/
- ffmpeg2theora http://www.v2v.cc/~j/ffmpeg2theora/
- png2theora from libtheora http://www.theora.org
- FFmpeg http://ffmpeg.org/
- APNG Assembler http://sourceforge.net/projects/apngasm/
- convert from ImageMagick http://www.imagemagick.org/

At least for pictures you need gnuplot and/or POV-Ray. The frames for all movie and animation output are rendered with POV-Ray.

2.5 References:

- Mohr, Daniel P. ; Stein, Ina ; Matzies, Thomas ; Knapek, Christina A.: Redundant robust topology optimization of truss. In: Optimization and Engineering (2013), 1-28. http://dx.doi.org/10.1007/s11081-013-9241-7
- Mohr, Daniel P. ; Stein, Ina ; Matzies, Thomas ; Knapek, Christina A.: Robust Topology Optimization of Truss with regard to Volume. In: arXiv - Mathematics, Optimization and Control (2012). http://arxiv.org/abs/1109.3782v2
- Mohr, Daniel P.: Redundante Topologieoptimierung. Neubiberg, Universitaet der Bundeswehr Muenchen, Fakultaet fuer Luft- und Raumfahrttechnik, Diss., Dezember 2011. http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:bvb:706-2664

2.6 copyright + license

Author: Daniel Mohr.

Date: 2013-11-09 (last change).

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

CHAPTER

THREE

TTRR_TOOLS.PY

ttrr_tools.py is a command line tool. You can get the help as usual:

```
$ ttrr_tools.py -h

usage: ttrr_tools.py [-h] -f file [file ...] [--new] [--version]
                     [--png_gnuplot] [--pdf_gnuplot] [--gnuplot_cylinder]
                     [--png_pov] [--ogv]
                     [-ogv_convert {ffmpeg2theora,png2theora,ffmpeg}] [--apng]
                     [--mng] [--gif] [--yuv] [--all_bars]
                     [--coordinate_system] [--small_coordinate_system]
                     [--show_result_table] [--show_results]
                     [-hist {log_asciiart,numbers,asciiart,all,none}]
                     [-ascii_hist_l lines] [-ascii_hist_c cols]
                     [--save_module] [--save_mat] [-x pixels] [-y pixels]
                     [-xm pixels] [-ym pixels] [-frames n] [-framerate n]
                     [-pov_angle a] [-pov_angle_m a] [-minfact mf]

ttrr - TrussTopologyRobustRedundant: ttrr is a software package to do topology
optimization of a truss with respect to robustness and redundancy. For more help
type "pydoc ttrr"

optional arguments:
  -h, --help            show this help message and exit
  -f file [file ...]    file(s) to process
  --new                 create only new pictures/movies. do not overwrite
                        existing ones.
  --version             show program's version number and exit
  --png_gnuplot         create png-pictures with gnuplot
  --pdf_gnuplot         create pdf-pictures with gnuplot
  --gnuplot_cylinder    Use gnuplot with creating cylinders as bars. This is
                        slow.
  --png_pov             create png-pictures with povray
  --ogv                 create ogv-movies with povray and a converter
  -ogv_convert {ffmpeg2theora,png2theora,ffmpeg}
                        Set the converter to generate the movie from the png-
                        pictures, which povray creates. To create a movie with
                        ffmpeg2theora it needs png2yuv. default: ffmpeg2theora
  --apng                create apng-animation with apngasm
  --mng                 create mng-animation with convert
  --gif                 create gif-animation with convert
  --yuv                 create YUV4MPEG stream format with png2yuv. This is an
                        uncompressed movie-stream. It is only useful for
                        further conversions.
  --all_bars            create pictures with all bars (structural universe)
  --coordinate_system   create pictures with a coordinate system
  --small_coordinate_system
                        create pictures with a small coordinate system
  --show_result_table   show results in a table
```

```
  --show_results        show results
  -hist {log_asciiart,numbers,asciiart,all,none}
                        Show in results histogram(s) of bar cross sections of
                        this type. default: log_asciiart
  -ascii_hist_l lines   Set in asciiart histogram the numbers of lines.
                        (default: 3)
  -ascii_hist_c cols    Set in asciiart histogram the numbers of cols.
                        (default: 100)
  --save_module         save with scipy.io.save_as_module data from file(s) to
                        "{.dat,.dir,.py}"-files
  --save_mat            save with scipy.io.savemat data from file(s) to
                        ".mat"-file(s)
  -x pixels             x resolution of the pictures (default: 1000)
  -y pixels             y resolution of the pictures (default: 1000)
  -xm pixels            x resolution of the movies (default: 800)
  -ym pixels            y resolution of the movies (default: 600)
  -frames n             the movie will be created from n frames (default: 250)
  -framerate n          set the framerate. the movie will be created with n
                        frames per second (default: 25)
  -pov_angle a          For rendering pictures by povray the view angle is set
                        to a (default: 33)
  -pov_angle_m a        For rendering animations by povray the view angle is
                        set to a (default: 40)
  -minfact mf           a truss with a cross section less than mf*max(s) is
                        ignored. max(s) is the maximum over all truss cross
                        sections. if mf=0 no truss is ignored. (default: 1e-6)

Author: Daniel Mohr
Date: 2013-11-09
License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007
```

CHAPTER

FOUR

EXAMPLE: BEN-TAL_NEMIROVSKI_PRIME_EXAMPLE_1997

Contents

This example shows you how `ttrr` is used. Some parts are unnecessary, but show the content/meaning of the concerning values or show same features/tools of `ttrr`. It's an example file!

This example is also available as `batch_ben-tal_nemirovski_prime_example_1997.py`.

This example describes the prime example of a truss from: Aharon Ben-Tal and Arkadi Nemirovski. Robust truss topology design via semidefinite programming. SIAM J. Optim., 7(4):991-1016, November 1997. ISSN 1052-6234 (print), 1095-7189 (electronic). http://dx.doi.org/10.1137/S1052623495291951 .

First start python and import necessary modules:

```
>>> import ttrr,numpy,scipy # we need ttrr and other great packages
```

4.1 design space

To define the design space:

```
>>> dim = 2 # dimension of the design space
>>> nelx = 3 # number of nodes in x direction
>>> nely = 2 # number of nodes in y direction
>>> kn = nelx*nely # number or total nodes
```

The definition of the grid:

```
>>> gx0 = 0.0 # meter
>>> gx1 = 2.0 # meter
>>> gy0 = 0.0 # meter
>>> gy1 = 1.0 # meter
```

Now we have a design space of nelx*nely=3*2=6 nodes. The design space is a rectangle of the points (x,y) with gx0 <= x <= gx1 and gy0 <= y <= gy1. We need the coordinates of all nodes:

```
>>> nodes = numpy.array([[ 0.0 , 0.0 ], # node 0 with the coordinates (0.0,0.0) = nodes[0,:]
                         [ 0.0 , 1.0 ], # node 1 with the coordinates (0.0,1.0) = nodes[1,:]
                         [ 1.0 , 0.0 ], # ...
                         [ 1.0 , 1.0 ],
                         [ 2.0 , 0.0 ],
                         [ 2.0 , 1.0 ]],dtype=numpy.float64) # the coordinates are float64 (doubl
```

We need a description of all trusses (from every node to every other one):

```
>>> bars = numpy.array([[0,2], # truss from node 0 with nodes[0,:] to node 2 with nodes[2,:]
                        [0,1], # truss from node 0 with nodes[0,:] to node 1 with nodes[1,:]
                        [0,3], # ...
                        [0,5],
                        [1,3],
                        [2,4],
                        [1,2],
                        [2,3],
                        [2,5],
                        [3,5],
                        [1,4],
                        [3,4],
                        [4,5]],dtype=numpy.uint16) # the number of nodes are uint16 (16-bit int
```

We do not need a truss from 0 to 4 or from 1 to 5, because it is the same as from 0 to 2 and from 2 to 4 or from 1 to 3 and from 3 to 5, respectively.

We can get this from, too (This is handy if you have more than 6 nodes!):

```
>>> [nodes,bars] = ttrr.barsnodesongrid2d(gx0,gx1,gy0,gy1,nelx,nely)
```

4.2 bearings

We want to fix the node 0 and 1 in every direction:

- 0 for node 0 in x direction
- 1 for node 0 in y direction
- 2 for node 1 in x direction
- 3 for node 1 in x direction

```
>>> mounting = numpy.array([0,1,2,3],dtype=numpy.uint16)
```

The truss from node 0 to node 1 is on both ends fixed. So this truss do not make sense. To reduce the number of design variables we can do it without this truss:

```
>>> [nodes,bars] = ttrr.barsnodesongrid2d(gx0,gx1,gy0,gy1,nelx,nely,mounting=mounting)
13  potenzielle Staebe gefunden; beruecksichtige 4 Lager
es bleiben 12 Staebe, die nicht nur gelagert sind
```

4.3 load cases

We have only 1 load case, but with “some” forces on different nodes:

```
>>> loads = scipy.sparse.lil_matrix((dim*kn,1),dtype=numpy.float64)
>>> loads[4,0] = -10000 # force in node 2 in x direction means 4 in the load case 0 with a value
>>> loads[7,0] = 10000 # force in node 3 in y direction means 7 in the load case 0 with a value o
>>> loads[9,0] = -10000 # force in -y direction with the magnitude 10000 N in node 4
>>> loads[10,0] = 10000 # force in x direction with the magnitude 10000 N in node 5
```

The magnitude of each force is only a scalar factor in the optimization problem. The commanding spot is the proportion of the forces among each other. The same holds for the material parameters.

4.4 calculation

We need a few parameters for the calculation:

```
>>> smax = 1 # the maximum allowed cross section in meter (here: not really a bound)
>>> name = "ben-tal_nemirovski_prime_example" # a name for the output
```

Some material parameter (The compressive strength sigmamin and the tensile strength sigmamax are only scalar factors in the calculation. The density density is only used to show the mass of the result.):

```
>>> sigma = 1e8 # Pa = 100 MPa = 100 N / mm^2 (yield strength for aluminium)
>>> sigmamin = - sigma # compressive strength
>>> sigmamax = sigma # tensile strength
>>> density = 2.7e3 # kg / m^3 (density for aluminium)
```

Now we can calculate the result:

```
>>> [s,w,lengths] = ttrr.ttrr(dim,nodes,bars,mounting,loads,smax,name=name,sigmamin=sigmamin,sigma
###### start ttrr ######
tmpdir: /tmp/ttrr_tmp__kA2eC
speichere nach ben-tal_nemirovski_prime_example.tar.bz2
### --- ### --- ### --- ### --- ### --- ### --- ###
starte: ttrr_calculate /tmp/ttrr_tmp__kA2eC
### ttrr ### gestartet ###
lese /tmp/ttrr_tmp__kA2eC/dim.bin
lese /tmp/ttrr_tmp__kA2eC/nodes.bin
lese /tmp/ttrr_tmp__kA2eC/bars.bin
lese /tmp/ttrr_tmp__kA2eC/mounting.bin
lese /tmp/ttrr_tmp__kA2eC/loads.bin
lese /tmp/ttrr_tmp__kA2eC/smax.bin
lese /tmp/ttrr_tmp__kA2eC/info.txt
lese /tmp/ttrr_tmp__kA2eC/redundancy.bin
lese /tmp/ttrr_tmp__kA2eC/opt.bin
lese /tmp/ttrr_tmp__kA2eC/sort.bin
lese /tmp/ttrr_tmp__kA2eC/sigmamin.bin
lese /tmp/ttrr_tmp__kA2eC/sigmamax.bin
lese /tmp/ttrr_tmp__kA2eC/density.bin
lese /tmp/ttrr_tmp__kA2eC/zlb.bin
lese /tmp/ttrr_tmp__kA2eC/zub.bin
ben-tal_nemirovski_prime_example: dim = 2 ; smax = 1.000000e+00 ; redundancy = 0 ; opt = 0 ; sort
sigmamin = -1.000000e+08 ; sigmamax = 1.000000e+08 ; density = 2.700000e+03
6 Knoten ; 12 Staebe ; 4 Lager ; 1 Lastfaelle
calculate geometry matrix and length of bars...geometry matrix (36 non-zero elements) and length
schreibe /tmp/ttrr_tmp__kA2eC/lengths.bin
schreibe /tmp/ttrr_tmp__kA2eC/lengths.txt
schreibe /tmp/ttrr_tmp__kA2eC/C.bin
schreibe /tmp/ttrr_tmp__kA2eC/C.txt
berechne freedofs...freedofs berechnet
schreibe /tmp/ttrr_tmp__kA2eC/freedofs.bin
```

```
schreibe /tmp/ttrr_tmp__kA2eC/freedofs.txt
beschraenke Geometriematrix (12x12) auf freie Knoten...Geometriematrix (8x12) auf freie Knoten bes
schreibe /tmp/ttrr_tmp__kA2eC/C_freedofs.bin
schreibe /tmp/ttrr_tmp__kA2eC/C_freedofs.txt
beschraenke Kraefte (12x1) auf freie Knoten...Kraefte (8x1) auf freie Knoten beschraenkt
schreibe /tmp/ttrr_tmp__kA2eC/loads_freedofs.bin
schreibe /tmp/ttrr_tmp__kA2eC/loads_freedofs.txt
erstelle Optimierungsproblem
berechne Zielfunktion...Zielfunktion berechnet
berechne Gleichungen...8 Gleichungen fuer 24 Unbekannte berechnet (26 Nicht-Null-Elemente)
schreibe /tmp/ttrr_tmp__kA2eC/Aeq.bin
schreibe /tmp/ttrr_tmp__kA2eC/Aeq.txt
schreibe /tmp/ttrr_tmp__kA2eC/beq.bin
schreibe /tmp/ttrr_tmp__kA2eC/beq.txt
berechne Unleichungen...24 Unleichungen fuer 24 Unbekannte berechnet (48 Nicht-Null-Elemente)
schreibe /tmp/ttrr_tmp__kA2eC/A.bin
schreibe /tmp/ttrr_tmp__kA2eC/A.txt
schreibe /tmp/ttrr_tmp__kA2eC/b.bin
schreibe /tmp/ttrr_tmp__kA2eC/b.txt
berechne lb und ub...lb und ub berechnet
schreibe /tmp/ttrr_tmp__kA2eC/lb.bin
schreibe /tmp/ttrr_tmp__kA2eC/lb.txt
schreibe /tmp/ttrr_tmp__kA2eC/ub.bin
schreibe /tmp/ttrr_tmp__kA2eC/ub.txt
A[23,23] = -1.000000e+08
erstelle Gesamtmatrix (32x24) aus 74 Nicht-Null-Elementen...erstellt; lade Gesamtmatrix...Gesamtma
Writing problem data to '/tmp/ttrr_tmp__kA2eC/ben-tal_nemirovski_prime_example.glpk.gz'...
69 lines were written
Writing problem data to '/tmp/ttrr_tmp__kA2eC/ben-tal_nemirovski_prime_example.prob.gz'...
141 lines were written
berechne...
opt == 0: glp_simplex
GLPK Simplex Optimizer, v4.45
32 rows, 24 columns, 74 non-zeros
Preprocessing...
32 rows, 24 columns, 74 non-zeros
Scaling...
 A: min|aij| =  4.472e-01  max|aij| =  1.000e+08  ratio =  2.236e+08
GM: min|aij| =  8.286e-01  max|aij| =  1.207e+00  ratio =  1.456e+00
EQ: min|aij| =  7.071e-01  max|aij| =  1.000e+00  ratio =  1.414e+00
Constructing initial basis...
Size of triangular part = 32
      0: obj =   0.000000000e+00  infeas =  1.740e+09 (0)
*    26: obj =   1.300000000e-03  infeas =  3.150e-12 (0)
*    29: obj =   8.000000000e-04  infeas =  8.078e-28 (0)
OPTIMAL SOLUTION FOUND
Writing basic solution to '/tmp/ttrr_tmp__kA2eC/ben-tal_nemirovski_prime_example.out.gz'...
Writing basic solution to '/tmp/ttrr_tmp__kA2eC/ben-tal_nemirovski_prime_example.sol.gz'...
58 lines were written
Zielfunktionswert: 0.000800
Gesamtvolumen: 0.000800
Gesamtgewicht: 2.160000
schreibe /tmp/ttrr_tmp__kA2eC/w.bin
schreibe /tmp/ttrr_tmp__kA2eC/w.txt
schreibe /tmp/ttrr_tmp__kA2eC/s.bin
schreibe /tmp/ttrr_tmp__kA2eC/s.txt
schreibe /tmp/ttrr_tmp__kA2eC/objectivefunctionvalue.bin
schreibe /tmp/ttrr_tmp__kA2eC/objectivefunctionvalue.txt
gebe Speicher wieder frei
### ttrr ### beendet ###
### --- ### --- ### --- ### --- ### --- ### --- ###
speichere nach ben-tal_nemirovski_prime_example.tar.bz2
###### end ttrr ######
```

As in "Aharon Ben-Tal and Arkadi Nemirovski. Robust truss topology design via semidefinite programming. SIAM J. Optim., 7(4):991-1016, November 1997. ISSN 1052-6234 (print), 1095-7189 (electronic). http://dx.doi.org/10.1137/S1052623495291951 ." we can get a robust solution; but here with a linear program as described in "Mohr, Daniel P. ; Stein, Ina ; Matzies, Thomas ; Knapek, Christina A.: Robust Topology Optimization of Truss with regard to Volume. In: arXiv - Mathematics, Optimization and Control (2012). http://arxiv.org/abs/1109.3782" (This time the output is omitted.):

```
>>> name = "ben-tal_nemirovski_prime_example_robust" # a name for the output
>>> [s,w,lengths] = ttrr.ttrr_robust(dim,nodes,bars,mounting,loads,smax,name=name,sigmamin=sigmam
```

More explicitly we can calculate (or even define) the perturbed forces:

```
>>> forces_disturbed = ttrr.compute_disturbed_forces(loads,dim,0.1,art=0)
###### compute_disturbed_forces
(kkn,nl) = (12,1); Knotenzahl: 6; Dimension: 2
###### compute_disturbed_forces
(kkn,nl) = (12,1); Knotenzahl: 6; Dimension: 2
######
max_force_undisturbed = 20000.000000
max_force_disturbed = 20784.609691
skaliere Kraefte
max_force_disturbed = 20000.000000
######
```

And do the computation with these (The output is omitted.):

```
>>> [s,w,lengths] = ttrr.ttrr(dim,nodes,bars,mounting,forces_disturbed,smax,name=name,sigmamin=si
```

For the same problem we can calculate a (1/2)-redundant solution as described in "Mohr, Daniel P. ; Stein, Ina ; Matzies, Thomas ; Knapek, Christina A.: Redundant robust topology optimization of truss. In: Optimization and Engineering (2013), 1-28. http://dx.doi.org/10.1007/s11081-013-9241-7" and in "Mohr, Daniel P.: Redundante Topologieoptimierung. Neubiberg, Universitaet der Bundeswehr Muenchen, Fakultaet fuer Luft- und Raumfahrttechnik, Diss., Dezember 2011. http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:bvb:706-2664":

```
>>> name = "ben-tal_nemirovski_prime_example_redundant" # a name for the output
>>> [s,w,lengths] = ttrr.ttrr_redundant(dim,nodes,bars,mounting,loads,smax,name=name,sigmamin=sig
```

4.5 visualization

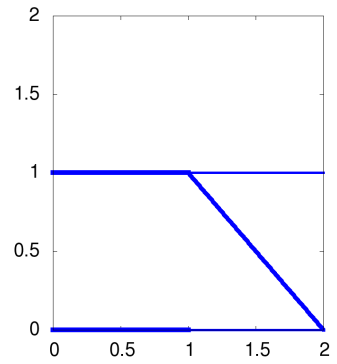
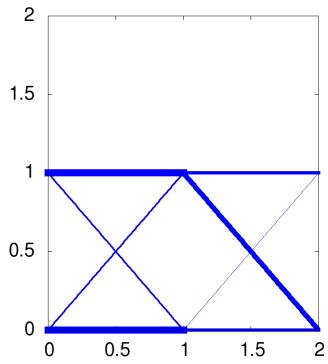
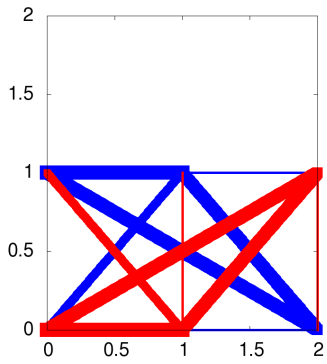
To get pictures, we can use *ttrr_tools.py* on the command line:

```
$ ttrr_tools.py --png_gnuplot -f ben-tal_nemirovski_prime_example.tar.bz2 ben-tal_nemirovski_prime
```

Or for short:

```
$ ttrr_tools.py --png_gnuplot -f ben-tal_nemirovski_prime_example*.tar.bz2
```

Now we get the example pictures:

optimal solution	robust solution	redundant solution
		

I think these pictures are the best way for our human being to understand the topologies. Keep in mind that the topology is not the sizing of parts of the structure. But the calculation needs the sizing values to benchmark the topology. So it is impossible to get a topology of a structure without the sizing values.

4.6 conclusion

The degree of freedom for this problem is very small. This means: The set of admissible solutions is small. So the optimization routine has only a small choice. The consequence is that in particular the result of the redundant solution is poor. But this file shows the functionality of ttrr and also the basic concept of robustness and redundancy. I hope you get the point.

CHAPTER

FIVE

EXAMPLE: 2_LOAD_CASES_3_DIM

Contents

This is a 3 dimensional example to show `ttrr`.

This example is also available as `batch_2_load_cases_3_dim.py`.

First start python and import necessary modules:

```
>>> import ttrr,numpy,scipy
```

5.1 design space

To define the design space:

```
>>> dim = 3 # dimension of the design space
>>> nelx = 3 # number of nodal points in x-direction
>>> nely = nelx # number of nodal points in y-direction
>>> nelz = nelx # number of nodal points in z-direction
```

To let you easily change the design space, we use here a calculation to define the grid. The measurement unit is meter:

```
>>> n = max(nelx,nely,nelz)
>>> gx0 = 0.0
>>> gx1 = (nelx-1)*1.0/(n-1)
>>> gy0 = 0.0
>>> gy1 = (nely-1)*1.0/(n-1)
>>> gz0 = 0.0
>>> gz1 = (nelz-1)*1.0/(n-1)
```

5.2 bearings

We want to fix all nodes on the ground. We can describe these area (All nodes will become fixed bearings):

```
>>> mountingarea = numpy.array([
...     [ gx0 , gx1 , gy0 , gy1 , gz0 , gz0 ]
...     ])
>>> mountingarea[:,[0,2,4]] = mountingarea[:,[0,2,4]] - 1e-6
>>> mountingarea[:,[1,3,5]] = mountingarea[:,[1,3,5]] + 1e-6
```

5.3 get the parameter

Now we can get the parameter for the calculation from barsnodesongrid3d. Here we can also use the optional parameter maxstablaenge to decide that bars longer than maxstablaenge should be ignored:

```
>>> maxstablaenge = 3 # maximum allowed length of a bar
>>> [nodes,bars,mounting] = ttrr.barsnodesongrid3d(gx0,gx1,gy0,gy1,gz0,gz1,nelx,nely,nelz,mountinc
mounting wird aus 1 Zeilen von mountingarea erstellt
9 Knoten sind gelagert; 27 Lager
302 potenzielle Staebe gefunden; beruecksichtige 27 Lager
es bleiben 274 Staebe, die nicht nur gelagert sind
274 potenzielle Staebe; beruecksichtige maximale Stablaenge 3.000000
es bleiben 274 Staebe, die kuerzer sind als 3.000000
```

We can look at the Parameter:

```
>>> nodes
array([[ 0. ,  0. ,  0. ],
       [ 0. ,  0.5,  0. ],
       [ 0. ,  1. ,  0. ],
       [ 0.5,  0. ,  0. ],
       [ 0.5,  0.5,  0. ],
       [ 0.5,  1. ,  0. ],
       [ 1. ,  0. ,  0. ],
       [ 1. ,  0.5,  0. ],
       [ 1. ,  1. ,  0. ],
       [ 0. ,  0. ,  0.5],
       [ 0. ,  0.5,  0.5],
       [ 0. ,  1. ,  0.5],
       [ 0.5,  0. ,  0.5],
       [ 0.5,  0.5,  0.5],
       [ 0.5,  1. ,  0.5],
       [ 1. ,  0. ,  0.5],
       [ 1. ,  0.5,  0.5],
       [ 1. ,  1. ,  0.5],
       [ 0. ,  0. ,  1. ],
       [ 0. ,  0.5,  1. ],
       [ 0. ,  1. ,  1. ],
       [ 0.5,  0. ,  1. ],
       [ 0.5,  0.5,  1. ],
       [ 0.5,  1. ,  1. ],
       [ 1. ,  0. ,  1. ],
       [ 1. ,  0.5,  1. ],
       [ 1. ,  1. ,  1. ]])
>>> bars
array([[ 0,  9],
       [ 0, 12],
       [ 0, 21],
```

```
       [ 0, 15],
       [ 0, 10],
       [ 0, 19],
       [ 0, 11],
       [ 0, 13],
       [ 0, 22],
       [ 0, 14],
       [ 0, 23],
       [ 0, 16],
       [ 0, 25],
       [ 0, 17],
       [ 9, 18],
       [ 9, 10],
       ...
       [17, 22]], dtype=uint16)
>>> mounting
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26])
>>> (kn,kr) = nodes.shape
>>> (sn,sr) = bars.shape
>>> (ln,) = mounting.shape
```

This means: We have kn=27 nodes and sn=274 bars. ln=27 node-directions are fixed. sr=2 is the number of nodes for a bar and kr=3 is the dimension of the design space.

5.4 load cases

The loads are stored in a sparse matrix. For every node we have dim directions. So we have dim*kn rows and 2 cols for the 2 load cases:

```
>>> loads = scipy.sparse.lil_matrix((dim*kn,2),dtype=numpy.float64) # 2 load cases
```

The number k1=19 of a node can be calculated from its coordinates. The direction r1=57 is in z-direction. The magnitude -100000.0 descibes that the direction is the negative z-direction:

```
>>> k1 = ttrr.nodenumberfromcoordinates(nodes,gx0,(gy0+gy1)/2,gz1) # position of the first force
>>> r1 = dim*k1 # direction of the first force
>>> loads[r1,0] = -100000.0 # Newton = ca. 1000 kg
>>> k1,r1
(19, 57)
```

The same for the other load case:

```
>>> k2 = ttrr.nodenumberfromcoordinates(nodes,gx1,(gy0+gy1)/2,gz1) # position of the second force
>>> r2 = dim*k2 # direction of the second force
>>> loads[r2,1] = 100000.0 # Newton = ca. 1000 kg
>>> k2,r2
(25, 75)
```

We can look at the result:

```
>>> print loads
  (57, 0)       -100000.0
  (75, 1)       100000.0
```

5.5 calculation

We need a few parameters for the calculation:

```
>>> smax = 1 # meter^2
>>> sigma = 1e8 # Pa = 100 MPa = 100 N / mm^2 (yield strength for aluminium)
>>> sigmamin = - sigma # compressive strength
>>> sigmamax = sigma # tensile strength
>>> density = 2.7e3 # kg / m^3 (density for aluminium)
>>> name="2_load_cases_3_dim"
```

Now we can do the classical topology optimization:

```
>>> [s,w,lengths] = ttrr.ttrr(dim,nodes,bars,mounting,loads,smax,name=name,sigmamin=sigmamin,sigma
```

To calculate a redundant solution we need (This takes about 47 seconds on a computer with 3.6 GHz):

```
>>> redundancy = 3
>>> name="2_load_cases_3_dim_redundant"
>>> [s,w,lengths] = ttrr.ttrr_redundant(dim,nodes,bars,mounting,loads,smax,name=name,sigmamin=sigm
```

5.6 visualization

To get pictures, we can use *ttrr_tools.py* on the command line. To get help:

```
$ ttrr_tools.py -h
```

We get quick and dirty pictures with gnuplot:

```
$ ttrr_tools.py --png_gnuplot -f 2_load_cases_3_dim*.tar.bz2
```

Nice pictures are rendered by povray:

```
$ ttrr_tools.py --png_pov -f 2_load_cases_3_dim*.tar.bz2
```

Because we are here viewing a 3 dimensional structure the 2 dimensional picture is insufficient. Therefore it is possible to get a movie animation:

```
$ ttrr_tools.py --ogv -f 2_load_cases_3_dim*.tar.bz2
```

If you only want a small animation you can do one of the following:

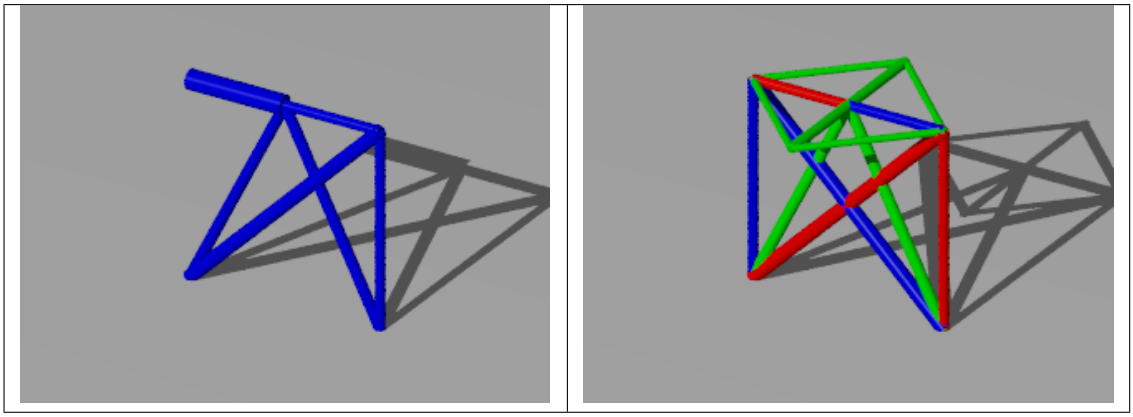
```
$ ttrr_tools.py --apng -xm 320 -ym 240 -f 2_load_cases_3_dim*.tar.bz2
$ ttrr_tools.py --mng -xm 320 -ym 240 -f 2_load_cases_3_dim*.tar.bz2
$ ttrr_tools.py --gif -xm 320 -ym 240 -f 2_load_cases_3_dim*.tar.bz2
```

You can also combine these options:

```
$ ttrr_tools.py -xm 320 -ym 240 -f 2_load_cases_3_dim*.tar.bz2 --png_pov --apng --mng --gif
```

The rendering by povray is time-consuming. Therefore it's done in parallel.

The above animation as apng:



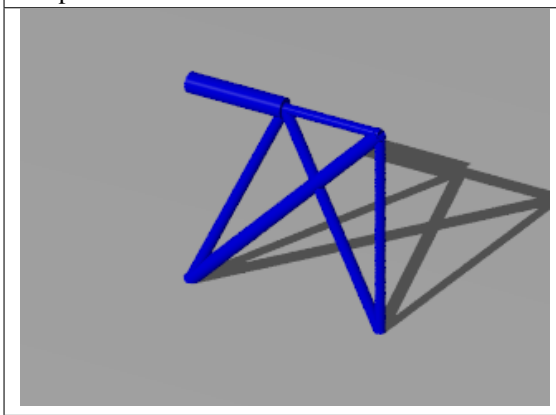
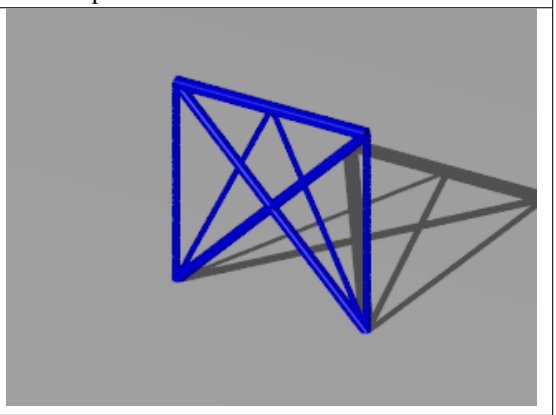
5.7 conclusion

Many people are wondering because the asymmetrical result. At last it's a symmetric problem. But we have a linear problem. Therefore there possibly exists more than one edge of the polytope as an optimal solution — the optimal value is the same. If we use an interior point algorithm we get a result between these edges.

The interior point algorithm of glpk has a big problem with scaling. Therefore we do a small scaling per hand (Note: This is only a scaling factor. The topology is the same.):

```
>>> name="2_load_cases_3_dim_interior"
>>> [s,w,lengths] = ttrr.ttrr(dim,nodes,bars,mounting,loads,smax,name=name,sigmamin=sigmamin,sigma
```

Now we can look at the result:

simplex	interior point
	

I hope you get the picture of the 2 edges of the polytope in this problem.

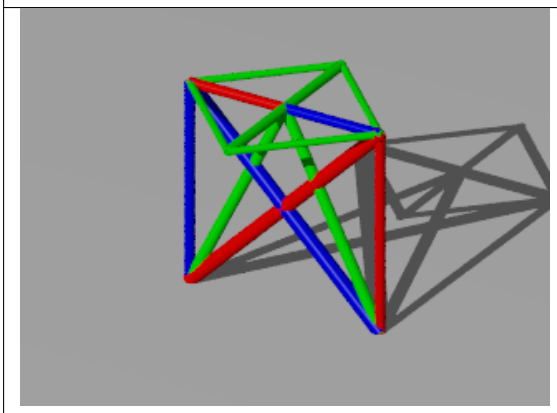
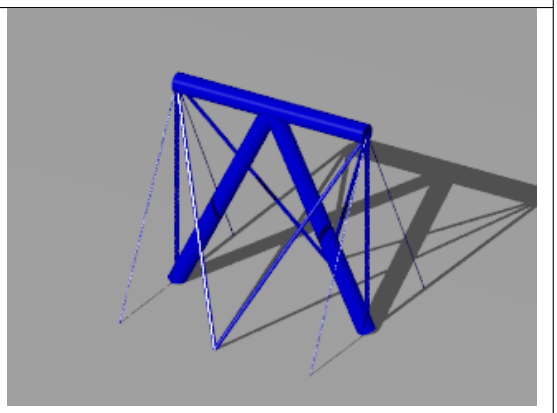
Another interesting point is the fact that we see a 2 dimensional structure in a 3 dimension space. Because the loads are only acting in one plane this is possible. Sure a small perturbation would destroy this behavior. Therefore it is not a practical solution. If you want a practical solution think of robust:

```
>>> name="2_load_cases_3_dim_robust"
>>> [s,w,lengths] = ttrr.ttrr_robust(dim,nodes,bars,mounting,loads,smax,name=name,sigmamin=sigmam
```

And an animation:

```
$ ttrr_tools.py -xm 320 -ym 240 -f 2_load_cases_3_dim_robust.tar.bz2 --png_pov --apng --mng --gif
```

The redundant solution is for a redundancy of 1/3. Every color represent a part of the structure. Every part is allow to drop out. The other parts are able to support the given loads. The interesting point is that the red or blue structure are not connected.

redundant	robust
	

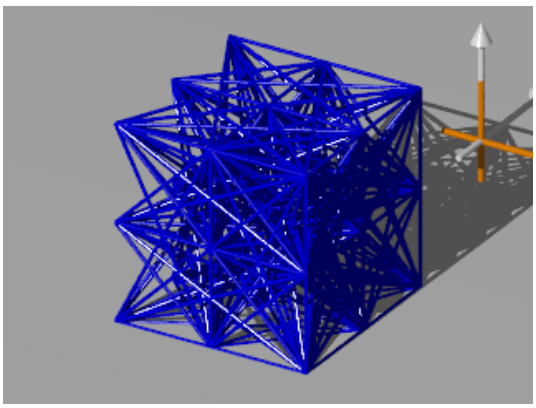
We can also compare the optimal values (This is the volume of the structures.):

optimal solution	redundant solution	optimal solution	robust solution
simplex	branch and bound	interior point	simplex
0.000875	0.001563	0.000875	0.001154

CHAPTER

SIX

EXAMPLE 3



This is a 3-dimensional example with the ground structure of 27 nodes $(a, b, c) \in \mathbb{R}^3$, $a, b, c \in \{1, 2, 3\}$ measured in meter and 274 potential bars between every two not fixed nodes – long bars which are located alongside several shorter bars are ignored. The supports are in the nodes $(1, a, b)$, $a, b \in \{1, 2, 3\}$ which are fixed in every direction. In the node $(3, 2, 1)$ acts a force of $4 \cdot 10^5$ N in the negative z-direction.

For a perturbation of $10\,\%$ in every direction of the force, and scaling to obtain the same maximal quantities as in the unperturbed case, our robust optimization method produces the robust optimal solutions. In contrast to the optimal solution the robust solution use the third dimension of the space.

6.1 Comparison of the volumes

	optimal	robust
volume:	6.0000e-03	6.5822e-03

Comparison of the volumes for different redundancies:

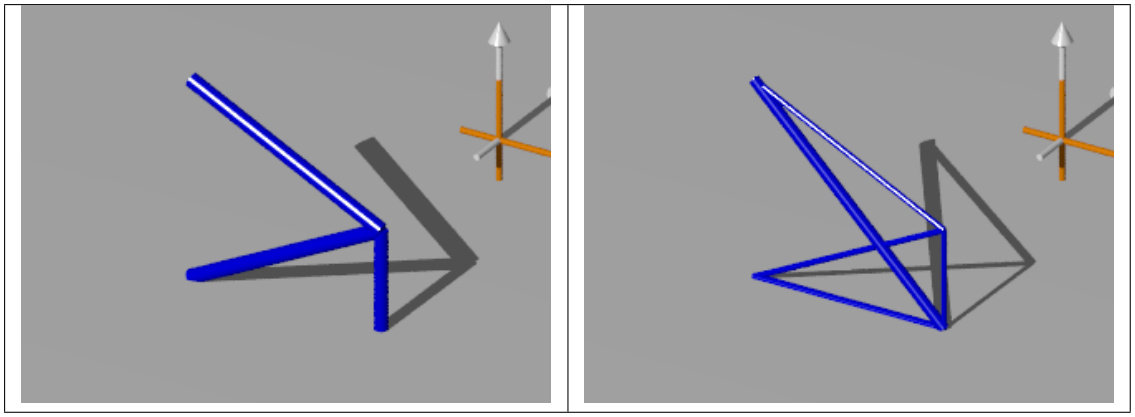
redundancy	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	$1-\frac{1}{2}$	$1-\frac{1}{3}$	$1-\frac{1}{4}$
volume:	8.6667e-03	9.5000e-03	1.2000e-02	1.2000e-02	1.9000e-02	2.6000e-02

Comparison of the volumes for different redundancies and robustness:

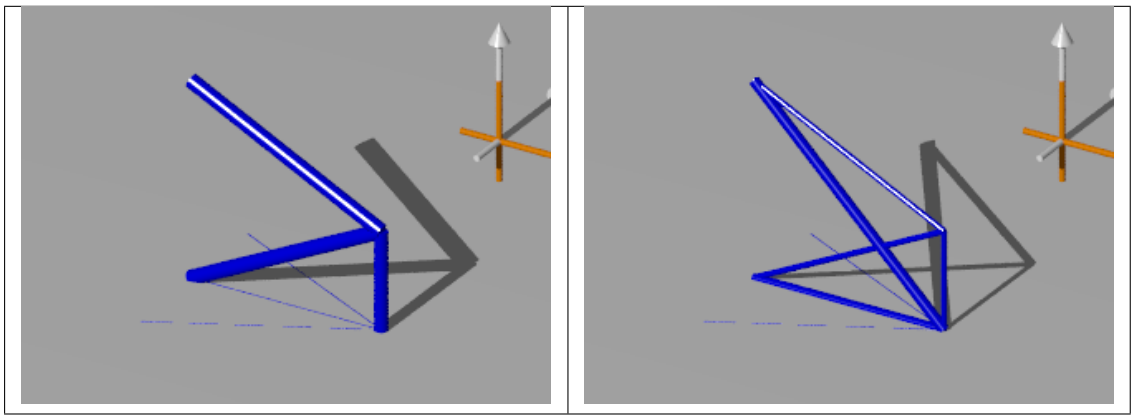
redundancy	$\frac{1}{3}$	$\frac{1}{2}$	$1-\frac{1}{2}$	$1-\frac{1}{3}$
volume:	1.2136e-02	1.5528e-02	1.5528e-02	2.4875e-02

6.2 Visualizations

Optimal solutions: (`script`)

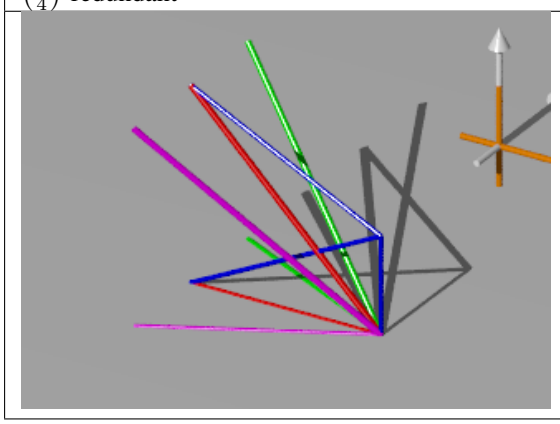


Robust solutions: (`script`)

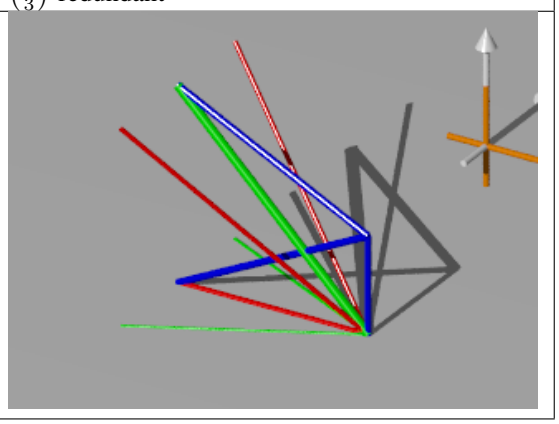


Redundant solutions: (`script`)

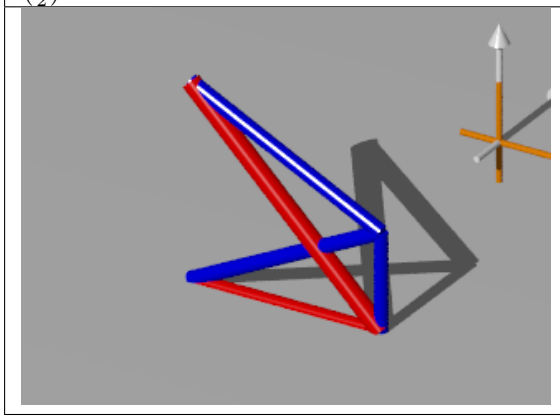
$\left(\frac{1}{4}\right)$-redundant



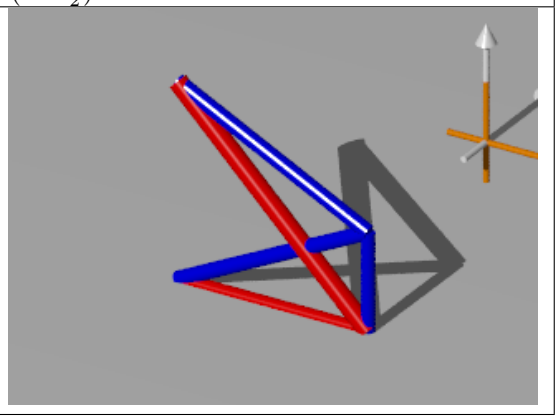
$\left(\frac{1}{3}\right)$-redundant

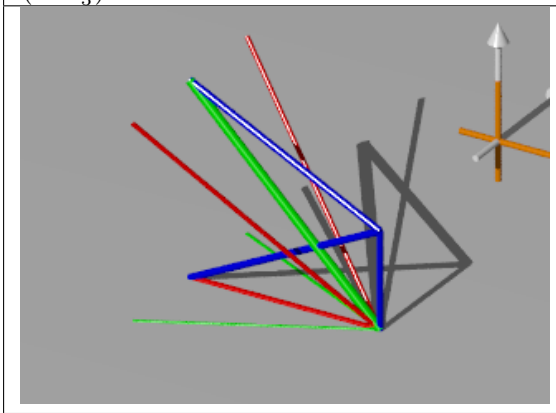
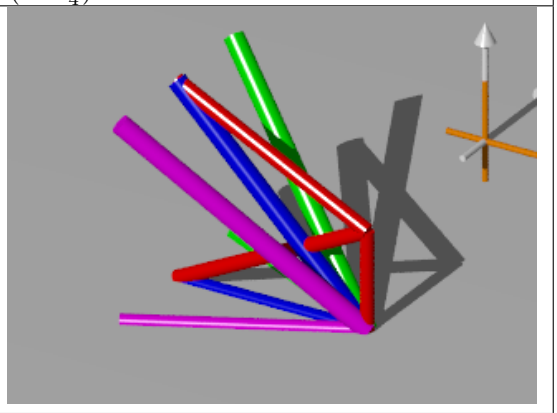


$\left(\frac{1}{2}\right)$-redundant

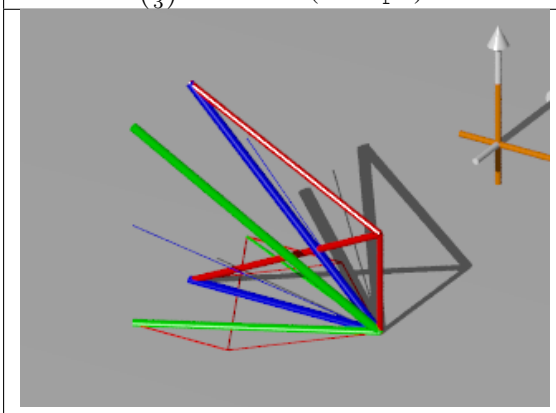
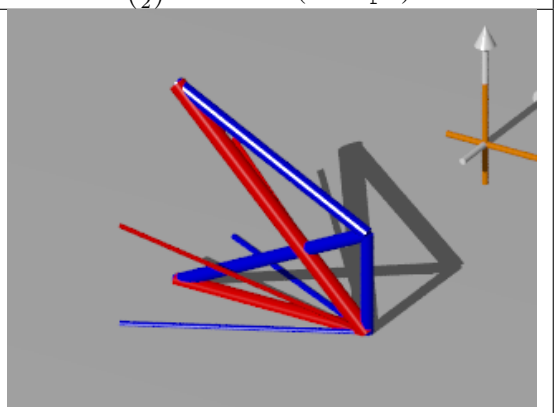
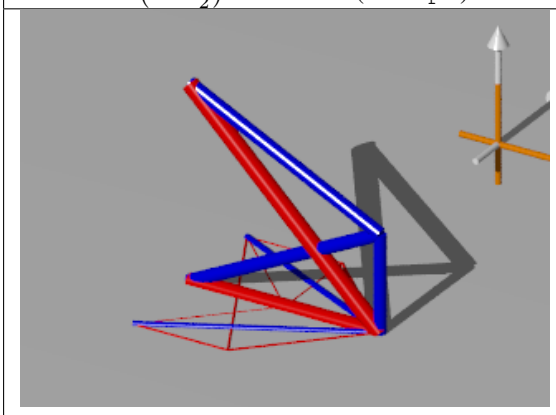
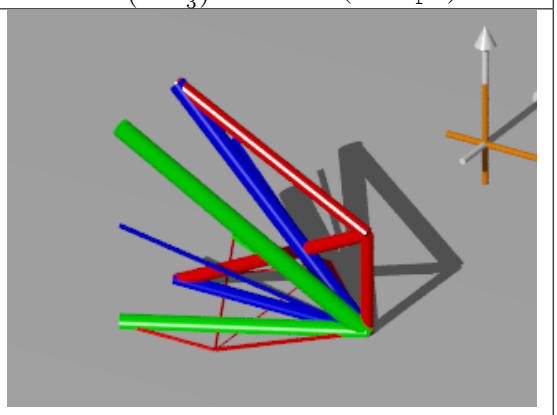


$\left(1-\frac{1}{2}\right)$-redundant



$\left(1-\frac{1}{3}\right)$-redundant	$\left(1-\frac{1}{4}\right)$-redundant
	

Robust and redundant solutions:

robust and $\left(\frac{1}{3}\right)$-redundant: (`script`)	robust and $\left(\frac{1}{2}\right)$-redundant: (`script`)
	
robust and $\left(1-\frac{1}{2}\right)$-redundant: (`script`)	robust and $\left(1-\frac{1}{3}\right)$-redundant: (`script`)
	

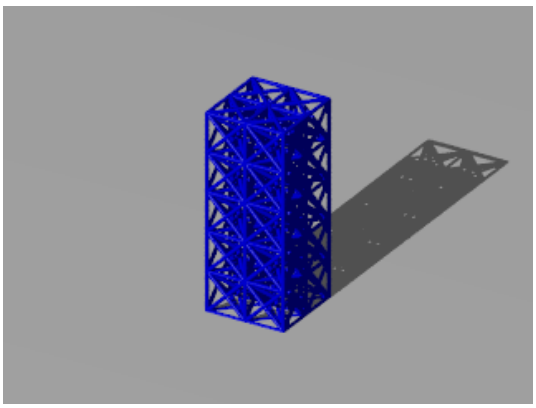
6.3 Python scripts

- `batch_example03.py`
- `batch_example03_robust.py`
- `batch_example03_redundant.py`
- `batch_example03_redundant_1div3_robust.py`
- `batch_example03_redundant_1div2_robust.py`
- `batch_example03_redundant_1minus1div2_robust.py`
- `batch_example03_redundant_1minus1div3_robust.py`

CHAPTER

SEVEN

EXAMPLE OF A MAST



In this example of a mast are 72 nodes $(a, b, c) \in \mathbb{R}^3$, $a, b \in \{0, \frac{1}{7}, \frac{2}{7}\}$, $c \in \{\frac{x}{7} : x = 0, 1, \ldots, 7\}$ measured in meter and 503 potential bars between every two not fixed nodes with a maximal distance of $\frac{\sqrt{3}}{7}$ — long bars which are located alongside several shorter bars are ignored. The supports are in the nodes $(a, b, 0)$, $a, b \in \{0, \frac{2}{7}\}$ which are fixed in every direction. In the node $\left(\frac{1}{7}, \frac{1}{7}, 1\right)$ acts a force of 10^5 N in the negative z-direction.

`The main problem is implemented as a script`. The other scripts are linked by the pictures.

For a perturbation of $50\,\%$ in every direction of the force, and scaling to obtain the same maximal quantities as in the unperturbed case, our robust optimization method produces the robust optimal solution.

7.1 Comparison of the volumes

	optimal	robust
volume:	1.4000e-03	3.4975e-03

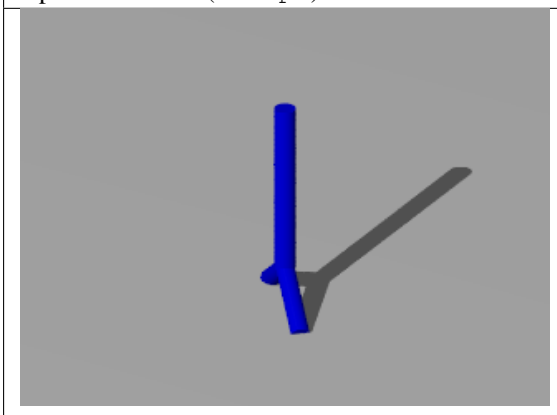
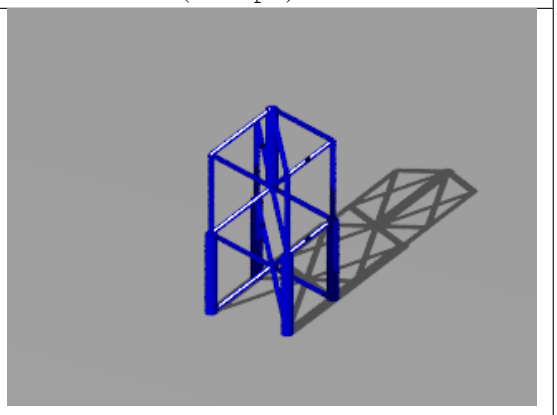
Comparison of the volumes for different redundancies:

redundancy	$\frac{1}{3}$	$\frac{1}{2}$	$1 - \frac{1}{2}$	$1 - \frac{1}{3}$
volume:	2.3000e-03	3.0000e-03	3.0000e-03	4.6000e-03

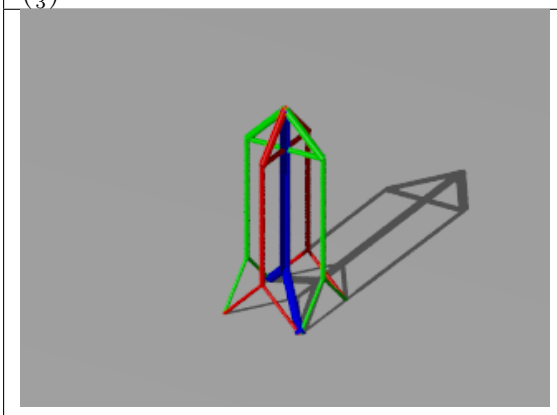
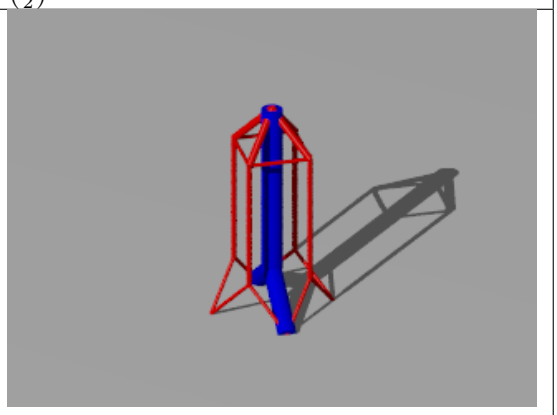
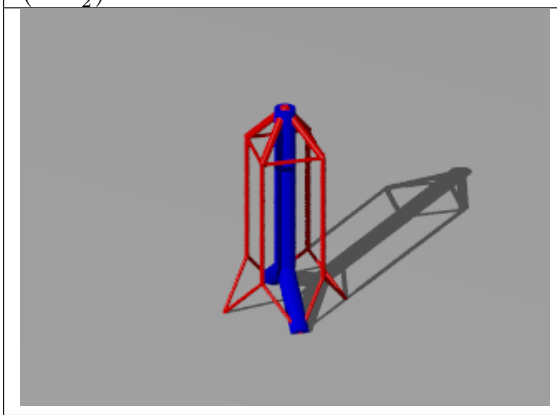
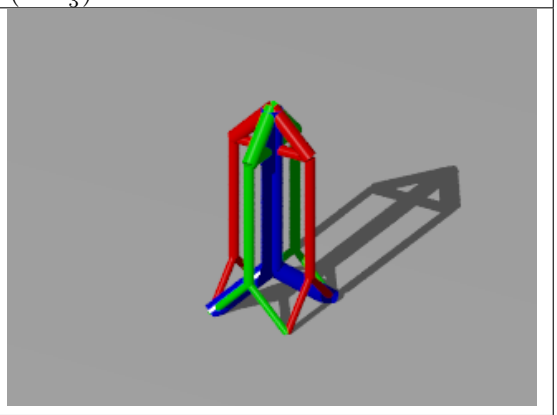
Comparison of the volumes for different redundancies and robustness:

redundancy	$\frac{1}{2}$	$1 - \frac{1}{2}$
volume:	8.0805e-03	8.0805e-03

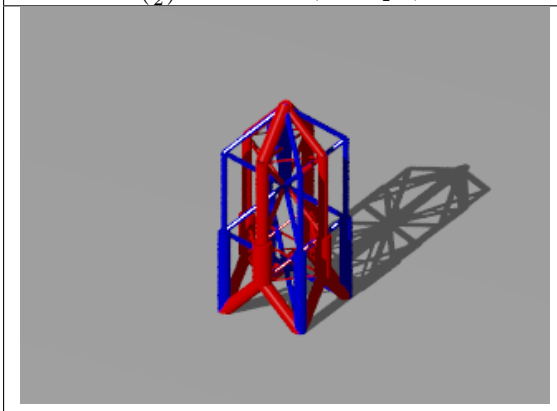
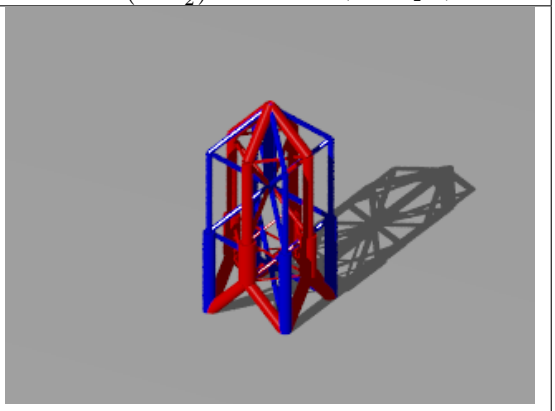
7.2 Visualizations

Optimal solution (`script`)	Robust solution (`script`)
	

Redundant solutions: (`script`)

$\left(\frac{1}{3}\right)$-redundant	$\left(\frac{1}{2}\right)$-redundant
	
$\left(1-\frac{1}{2}\right)$-redundant	$\left(1-\frac{1}{3}\right)$-redundant
	

Robust and redundant solutions:

robust and $\left(\frac{1}{2}\right)$-redundant: (`script`)	robust and $\left(1-\frac{1}{2}\right)$-redundant: (`script`)
	

7.3 Python scripts

- `mast_aufgabenstellung.py`
- `batch_example05.py`
- `batch_example05_robust.py`
- `batch_example05_redundant.py`
- `batch_example05_redundant_1div2_robust.py`
- `batch_example05_redundant_1minus1div2_robust.py`

CHAPTER

EIGHT

MATHEMATICAL BACKGROUND

The software package `ttrr` emerged from the scientific research for [Mohr2011], [MohrSteinMatziesKnapek2012] and [MohrSteinMatziesKnapek2013].

First, I started with MatLab http://www.mathworks.de/ in my PhD research, but the problems were too hard for this great software package. Furthermore, MatLab does not support the solving of mixed integer programs on its own. Therefore I was looking for something else: I found GLPK http://www.gnu.org/software/glpk/ and Python http://www.python.org/ — both are great software!

Because it was simple, I did the visualization with MatLab http://www.mathworks.de/ in my PhD-thesis [Mohr2011]. To write `ttrr`, I do not want to use a commercial back end for the visualization or moreover point you to such a software. But if you like, you can do it on your own.

Contents

8.1 General Concept of Topology Optimization of Truss

Truss topology optimization has its root in [Michell1904].

The analytically stated and known (cf. [Przemieniecki1968]; [Marti2003]; [BendsoeSigmund2004]) linear problem of the topology optimization of a truss with respect to its volume (plastic design) is given by the objective function to be minimized

$$\min_{s,w} l^T s$$

with the bar lengths $l \in \mathbb{R}^n$ subject to the equilibrium condition

$$Cw = f$$

with the reduced geometry matrix $C \in \mathbb{R}^{m \times n}$, the inner bar forces $w \in \mathbb{R}^n$ and the reduced applied external loads $f \in \mathbb{R}^m$. Further given are the constraints

$$\sigma_- s \leq w \leq \sigma_+ s$$

for the linear elasticity by — for all bars equal — stress limits $0 > \sigma_- \in \mathbb{R}$ and $0 < \sigma_+ \in \mathbb{R}$ (e. g. yield points for pressure and tension of the material all bars are made of), and the box constraint

$$0 \leq s \leq s_{\max}$$

for the cross sections of the bars $s \in \mathbb{R}^n$ with the given maximal bar cross section $s_{\max} \in \mathbb{R}^n$. [1]

Here w is seen as a design variable; otherwise we have $s(w)$ as function and therefore no linear program.

If more than one load (e. g. the set F of loads) is to be considered, one ends up at the known multiple load case. But how is it performed? How can we put more than one load in the above linear program? It is impossible! But we can ask for a robust solution — this means we consider the worst case: A solution for all $f^{(i)} \in F$ is searched. Now we really end up at the well known multiple load case: [2]

$$\begin{aligned} \text{obj. func.: } V(s) = l^T s &\to \min_{s,w} \\ \text{s.t.: } \forall f^{(i)} \in F &: \\ C w^{(i)} &= f^{(i)} \in \mathbb{R}^m \\ \sigma_- s \leq w^{(i)} &\leq \sigma_+ s \\ 0 \leq s &\leq s_{\max} \end{aligned}$$

All this is done by `ttrr.ttrr()`.

8.2 Robust Topology Optimization of Truss

For robust optimization see [Ben-TalElGhaouiNemirovski2009].

The implementation in `ttrr.ttrr_robust()` follows [MohrSteinMatziesKnapek2012]. This means if perturbed loads $\tilde{F}$ are given, they are composed by some perturbations of the unperturbed force f. Due to the linear programming we can restrict ourselves to the extreme points of $\tilde{F}$.

If we start the other way round with an additional assumption, it becomes clear: Let f be a given force with a perturbation of 10%. Then we are interested in a solution for all $\tilde{f} \in \{\tilde{f} : 0.9f \leq \tilde{f} \leq 1.1f\}$. Obviously the extreme points are $0.9f$ and $1.1f$. Therefore a solution for all loads in the set $F := \{0.9f; 1.1f\}$ is searched.

So robust optimization in topology optimization is the same as the multiple load case with appropriate load cases. For F a polyhedron a finite number of extreme points is given. And therefore `ttrr.ttrr()` can be recycled.

In addition `ttrr.ttrr_robust()` provides features to calculate the extreme points of simple sets of perturbed forces.

8.3 Redundant Topology Optimization of Truss

The basic concept of topology optimization with regard to redundancy is described in [Mohr2011] and in [MohrSteinMatziesKnapek2013].

Redundant Topology Optimization of Truss means we are looking for the best topology of a truss with a redundancy as to the falling out of some parts of the structure. The dream is a structure which is functional even if a significant damage occurs to an arbitrary set of bars.

You should think of RAID [3]. [PattersonGibsonKatz1987]

For example, a RAID 1 is a bunch of discs and every disc does the work. This means every other one can fail. Following [Mohr2011] this is called "1-1/n"-redundancy (c.f. [Shannon1948] [PattersonGibsonKatz1987]). Transferred to structures this means we are looking for n different structures in the same design space fulfilling the task to sustain the load(s).

[1] The maximal cross section is no limitation — it's a feature! It provides the possibility to limit the cross section; but if you do not want this set a huge upper limit — but an adequate limit helps the optimization algorithms.

[2] Sorry for the compactness, but you can look at the literature all over the world. :-)

[3] RAID is an acronym for Redundant Array of Independent Disks. http://en.wikipedia.org/wiki/RAID

Let $s_i \in \mathbb{R}^N$ be the cross sections of the bars for the structure i. The sum $s = \sum_{i=1}^{n} s_i$ describes the cross sections of the bars for the complete structure. The complete structure is composed of the n structures s_i. With the selecting variables $x_i \in \{0;1\}^N$ we can now formulate the problem as a mixed integer linear program: [4]

$$\begin{aligned}
\text{obj. func.: } V(s_1, s_2, \ldots, s_n) &= \sum_{i=1}^{n} l^T s_i \to \min_{x,w,s} \\
\text{s.t.: } \forall i &= 1, 2, \ldots, n : \\
& s_i \le s_{\max} x_i \in \mathbb{R}^N \\
& \forall k = 1, 2, \ldots, n_l : \\
& \quad C w_i^{(k)} = f^{(k)} \in \mathbb{R}^m \\
& \quad \sigma_- s_i \le w_i^{(k)} \le \sigma_+ s_i \\
& \sum_{i=1}^{n} x_i \le 1 \\
& x \in \{0;1\}^{nN}; \qquad 0 \le s
\end{aligned}$$

From $x \in \{0;1\}^{nN}$ follows $0 \le \sum_{i=1}^{n} x_i$ and $s \le s_{\max}$.

This is done by `ttrr.ttrr_redundant()` for *redundancy* = $-n \le -2$.

A RAID 5 (c.f. [PattersonGibsonKatz1987]) is more complex. It can be described as a simple code. To transfer this concept to structures we only need to know the behavior. A RAID 5 is a bunch of discs. Every single disc may fail and the others do the job together. Following [Mohr2011] this is called "1/n"-redundancy (c.f. [Shannon1948] [PattersonGibsonKatz1987]). Back to structures this means we are looking for n different structures in the same design space. Every single structure can fall out and the other ones should sustain the load(s) together.

Let $s_i \in \mathbb{R}^N$ be the cross sections of the bars for the structure i. As before $s = \sum_{i=1}^{n} s_i$ describes the cross sections of the bars for the complete structure. The complete structure s is composed of the n structures s_i. If a structure i fall out, the resulting structure is $\bar{s}_i = \sum_{j=1}^{n} s_j - s_i$. As mentioned before, this damaged structure should sustain the load(s).

With the selecting variables $x_i \in \{0;1\}^N$ we can now formulate the problem as a mixed integer linear program: [5]

$$\begin{aligned}
\text{obj. func.: } V(s_1, s_2, \ldots, s_n) &= \sum_{i=1}^{n} l^T s_i \to \min_{x,w,s} \\
\text{s.t.: } \forall i &= 1, 2, \ldots, n : \\
& s_i \le s_{\max} x_i \in \mathbb{R}^N \\
& \forall k = 1, 2, \ldots, n_l : \\
& \quad C w_i^{(k)} = f^{(k)} \in \mathbb{R}^m \\
& \quad \sigma_- \bar{s}_i \le w_i^{(k)} \le \sigma_+ \bar{s}_i \\
& \sum_{i=1}^{n} x_i \le 1 \\
& x \in \{0;1\}^{nN}; \qquad 0 \le s \\
\text{with: } \bar{x}_i &= \sum_{j=1, j\ne i}^{n} x_j; \qquad \bar{s}_i = \sum_{j=1, j\ne i}^{n} s_j
\end{aligned}$$

Trivially $0 \le \sum_{i=1}^{n} \bar{x}_i \le 1$ holds and therefore $0 \le \bar{x}_i \le 1$. From $x \in \{0;1\}^{nN}$ follows $0 \le \sum_{i=1}^{n} x_i$ and $s \le s_{\max}$. The damaged structures are also well defined:

$$\bar{s}_i = \sum_{j=1, j\ne i}^{n} s_j \le s_{\max} \sum_{j=1, j\ne i}^{n} x_j = s_{\max} \bar{x}_i \le s_{\max}$$

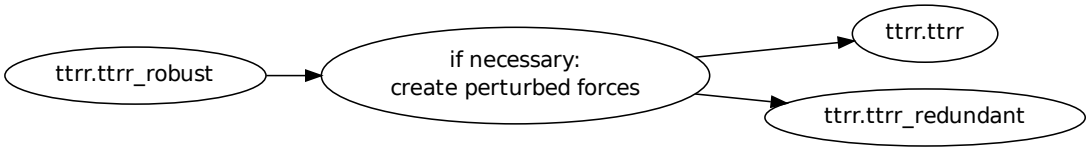
[4] MIP: mixed integer problem; MILP: mixed integer linear program. Because MIP in general is too time consuming everybody is only thinking of MILP and often calls it MIP.

Clearly $0 \leq \bar{s}_i$ holds. So we get for the damaged structures $0 \leq \bar{s}_i \leq s_{\max}$ and for the complete structure $0 \leq s \leq s_{\max}$.

This is done by `ttrr.ttrr_redundant()` for redundancy = $n \geq 2$.

8.4 Robust Redundant Topology Optimization of Truss

Robust redundant topology optimization is nothing more than redundant topology optimization with appropriate load cases. This concept of redundant robust topology optimization is described in [MohrSteinMatziesKnapek2013].



8.5 Uniqueness of the solution

For a linear problem possibly more than one edge of the polyhedron exists as a solution. So the solution of a linear problem is not unique — but the optimal object value is.

Let us consider the small example:

$$\begin{aligned} \text{obj. func.: } z^T x &:= \begin{pmatrix} 1 & 1 \end{pmatrix} x \to \max_x \\ \text{s.t.: } Ax &:= \begin{pmatrix} 1 & 1 \end{pmatrix} x \leq b := 1 \\ x &\geq 0 \end{aligned}$$

A visualization is given in Fig. *Visualization of the simple LP*.

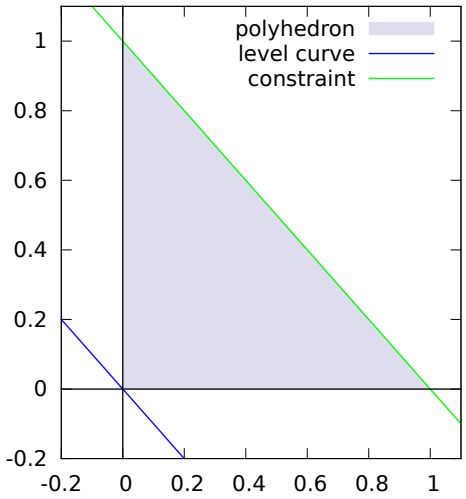
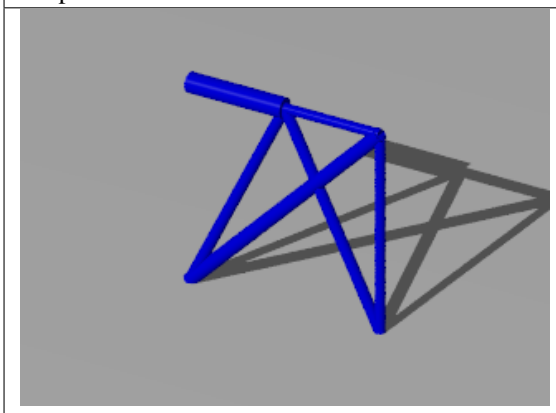
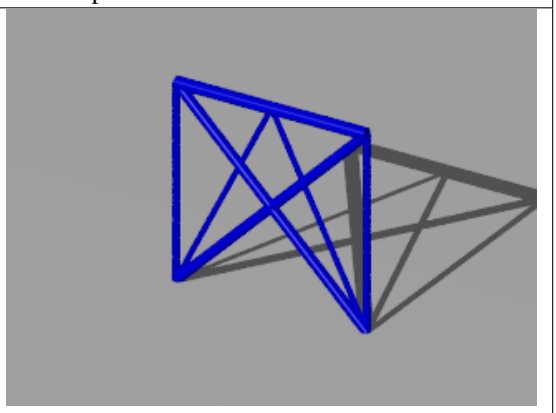


Figure 8.1: Visualization of the simple LP
In the visualization we see a level curve for $z^T x = 0$ and the polyhedron $\{x \in \mathbb{R}^2 : Ax \leq b \wedge x \geq 0\}$ bounded by the constraints.

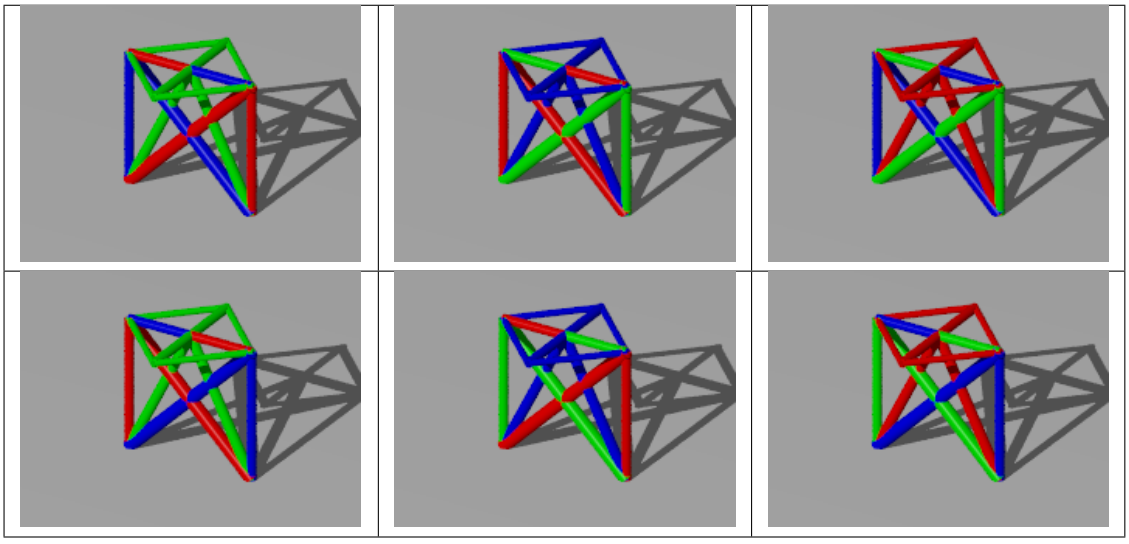
Without a lengthy calculation we can see that the optimal value of the objective function $z^T x$ is 1. The edges $(0, 1)$ and $(1, 0)$ are feasible, optimal and have both the same value of the objective function — the same for all points between these edges.

So in general you cannot expect a unique solution for a linear program. The simplex algorithm will find an edge as a solution. An interior point algorithm will typically find a solution between edges which are optimal. The following two pictures from *Example: 2_load_cases_3_dim* show this behavior — both are optimal for the same problem.

simplex	interior point
	

For the redundant topology optimization we have in every case more than one solution. For example let $\tilde{x}, \tilde{w}, \tilde{s}$ with $\tilde{s} = (\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n)$ be a solution, then $(\tilde{s}_2, \tilde{s}_1, \ldots, \tilde{s}_n)$ is one, too.

For the example *Example: 2_load_cases_3_dim* for a "1/3"-redundancy this is shown in the following table:



To mostly overcome this behavior we can require an order of the volumes of the structures:

$$l^T s_1 \leq l^T s_2 \leq \ldots \leq l^T s_n$$

This is implemented in `ttrr.ttrr_redundant()` with the flag "sort".

8.6 Scaling Factors

To look at scaling behavior of the above linear programs let us consider the following structure:

$$\begin{aligned} \text{obj. func.: } V(s) = l^T s &\to \min_{s,w} \\ Cw &= f \\ \sigma_- s \leq w &\leq \sigma_+ s \\ 0 \leq s &\leq s_{\max} \end{aligned}$$

8.6.1 Material, Load and Design Space

The material parameters for the topology optimization of truss given to `ttrr.ttrr()`, `ttrr.ttrr_robust()` or `ttrr.ttrr_redundant()` are:

- $s_{\max}$: the maximal allowed cross section of a bar
- σ_- : lower limit for the allowed stress, e.g. compressive strength
- σ_+ : upper limit for the allowed stress, e.g. tensile strength
- density : density

The density is only used to calculate the mass of the structure out of its volume.

$s_{\max}$ is necessary for `ttrr.ttrr_redundant()` and gives an additional feature to `ttrr.ttrr()` and `ttrr.ttrr_robust()`. If its not wanted a huge value would disable it. But in any case it gives with $0 \leq s \leq s_{\max}$ bounding values. So we have a closed feasible region.

σ_- and σ_+ are really used material parameters. Typical topology optimization is a preliminary design. Therefore it must not be totally accurate to the praxis. Typically $\sigma_- = -\sigma_+ < 0$ is chosen. So we can assume $\sigma_- = -\sigma_+ = -\sigma$ for some $\sigma > 0$.

$$\begin{aligned} \text{obj. func.: } V(s) = l^T s &\to \min_{s,w} \\ Cw &= f \\ -\sigma s \leq w &\leq \sigma s \\ 0 \leq s &\leq s_{\max} \end{aligned}$$

Easily to see the following program results in an equivalent optimum. With $\tilde{s} = \sigma s$ we get:

$$\begin{aligned} \text{obj. func.: } V(\tilde{s}) = l^T \tilde{s} &\to \min_{\tilde{s},w} \\ Cw &= f \\ -\tilde{s} \leq w &\leq \tilde{s} \\ 0 \leq \tilde{s} &\leq \frac{1}{\sigma} s_{\max} \end{aligned}$$

This is only a scaling. The magnitude of the loads is only a scaling, too. With $\tilde{w} = \alpha w$ for some $\alpha > 0$ we get:

$$\begin{aligned} \text{obj. func.: } V(\tilde{s}) = l^T \tilde{s} &\to \min_{\tilde{s},\tilde{w}} \\ C\tilde{w} &= \alpha f \\ -\alpha\tilde{s} \leq \tilde{w} &\leq \alpha\tilde{s} \\ 0 \leq \tilde{s} &\leq \frac{1}{\sigma} s_{\max} \end{aligned}$$

And furthermore with $\hat{s} = \alpha\tilde{s} = \alpha\sigma s$ the following program is equivalent, too:

$$\begin{aligned} \text{obj. func.: } V(\hat{s}) = l^T\hat{s} &\to \min_{\hat{s},\tilde{w}} \\ C\tilde{w} &= \alpha f \\ -\hat{s} \le \tilde{w} &\le \alpha\hat{s} \\ 0 \le \hat{s} &\le \frac{1}{\alpha\sigma}s_{\max} \end{aligned}$$

Keep in mind it is not required to change the objective function. For $\beta > 0$ the minimum of the functions $g(x)$ and $\beta g(x)$ is reached at the same point.

The size of the design space is not found directly in the linear program. In the geometry matrix C occur only the angles of the bars. So the magnitude of the size of the design space in meter or millimeter will result in the same optimal structure, because l is only in the objective function and will be changed by a scalar factor.

8.6.2 Numeric

A scaling of

$$\begin{aligned} \text{obj. func.: } V(s) = l^T s &\to \min_{s,w} \\ Cw &= f \\ \sigma_- s \le w &\le \sigma_+ s \\ 0 \le s &\le s_{\max} \end{aligned}$$

do not change the (relative) condition of the problem. But it can change the absolute condition and therefore helps the computer by the calculation.

To do so the functions `ttrr.ttrr()`, `ttrr.ttrr_robust()` or `ttrr.ttrr_redundant()` have the parameters $\alpha =$ scale_sigma and $\beta =$ scale_forces. To get the picture let us see how they transform the above problem:

$$\begin{aligned} \text{obj. func.: } V(s) = l^T s &\to \min_{s,w} \\ C\beta w &= \beta f \\ \alpha\sigma_- \frac{\beta}{\alpha}s \le \beta w &\le \alpha\sigma_+ \frac{\beta}{\alpha}s \\ 0 \le s &\le s_{\max} \end{aligned}$$

Now the calculation will be done with $\tilde{s} = \frac{\beta}{\alpha}s$, $\tilde{w} = \beta w$, $\tilde{f} = \beta f$, $\tilde{\sigma}_- = \alpha\sigma_-$, $\tilde{\sigma}_+ = \alpha\sigma_+$ and $\tilde{s}_{\max} = \frac{\beta}{\alpha}s_{\max}$:

$$\begin{aligned} \text{obj. func.: } V(\tilde{s}) = l^T\tilde{s} &\to \min_{\tilde{s},\tilde{w}} \\ C\tilde{w} &= \tilde{f} \\ \tilde{\sigma}_-\tilde{s} \le \tilde{w} &\le \tilde{\sigma}_+\tilde{s} \\ 0 \le \tilde{s} &\le \tilde{s}_{\max} \end{aligned}$$

Afterwards $\tilde{s}$ and $\tilde{w}$ will be transformed back.

For a maximum magnitude of 1 of an element from $\tilde{f}$ the absolute condition of the equation will be a little bit better. For a maximum magnitude of 1 of an element from $\tilde{\sigma}_-$ and from $\tilde{\sigma}_+$ the condition of the inequation will be significant better. If $\alpha = \beta$ the design variables s and $s_{\max}$ resp. will not be changed.

CHAPTER

NINE

README: TTRR - TRUSSTOPOLOGY-ROBUSTREDUNDANT

ttrr is a software package to do topology optimization of a truss with respect to robustness and redundancy.

9.1 install

see *INSTALL: ttrr - TrussTopologyRobustRedundant*

9.2 literature

- Mohr, Daniel P. ; Stein, Ina ; Matzies, Thomas ; Knapek, Christina A.: Redundant robust topology optimization of truss. In: Optimization and Engineering (2013), 1-28. http://dx.doi.org/10.1007/s11081-013-9241-7
- Mohr, Daniel P. ; Stein, Ina ; Matzies, Thomas ; Knapek, Christina A.: Robust Topology Optimization of Truss with regard to Volume. In: arXiv - Mathematics, Optimization and Control (2012). http://arxiv.org/abs/1109.3782v2
- Mohr, Daniel P.: Redundante Topologieoptimierung. Neubiberg, Universitaet der Bundeswehr Muenchen, Fakultaet fuer Luft- und Raumfahrttechnik, Diss., Dezember 2011. http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:bvb:706-2664

9.3 copyright + license

Author: Daniel Mohr.

Date: 2013-11-09 (last change).

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

CHAPTER

TEN

INSTALL: TTRR - TRUSSTOPOLOGY-ROBUSTREDUNDANT

10.1 info

ttrr is a software package to do topology optimization of a truss with respect to robustness and redundancy.

10.2 before you install

Before you install, make sure to have:

- Python http://www.python.org/
- NumPy http://numpy.scipy.org/ http://pypi.python.org/pypi/numpy
- SciPy http://www.scipy.org/ http://pypi.python.org/pypi/scipy
  - and also for NumPy/SciPy: BLAS and LAPACK libraries
- GLPK http://www.gnu.org/software/glpk/

To get graphical output (picture, movie, animation) you need some of the following programs:

- gnuplot http://www.gnuplot.info/
- POV-Ray http://www.povray.org/
- png2yuv from MJPEG http://mjpeg.sourceforge.net/
- ffmpeg2theora http://www.v2v.cc/~j/ffmpeg2theora/
- png2theora from libtheora http://www.theora.org
- FFmpeg http://ffmpeg.org/
- APNG Assembler http://sourceforge.net/projects/apngasm/
- convert from ImageMagick http://www.imagemagick.org/

At least for pictures you need gnuplot and/or POV-Ray. The frames for all movie and animation output are rendered with POV-Ray.

There is also a small extra command to check for availability of necessary python modules:

```
python setup.py check_modules
```

If you want to use this complete software you should have no modules which are not available.

10.2.1 Python3

If you use Python3, you must convert to Python3 via 2to3:

```
tar xzf ttrr-*.tar.*
cd ttrr-*/
2to3 -w .
```

10.3 install

10.3.1 global-install

To install this software global to / the following steps are to perform:

```
tar xzf ttrr-*.tar.*
cd ttrr-*/
python setup.py build_c_scripts
python setup.py install
```

Please recognize the extra step *python setup.py build_c_scripts* here.

10.3.2 home-install

To install this software to your $HOME the following steps are to perform:

```
tar xzf ttrr-*.tar.*
cd ttrr-*/
python setup.py build_c_scripts
python setup.py install --home=~
```

Please recognize the extra step *python setup.py build_c_scripts* here.

10.4 hints

- GLPK 4.38 and 4.40 do not work.

  GLPK 4.43 and 4.44 and 4.45 and 4.47 work.

  GLPK 4.52.1 gives other binary results compared to "results.tar.bz2". But these are only numerical rounding errors.

- Keep in mind to have the right pathes.

  For the above installation to $HOME the software installs in:

  ```
  ~/bin
  ~/lib/python
  ```

  Please make sure to have these pathes in $PATH and $PYTHONPATH, respectively. For example:

  ```
  export PATH=$PATH:~/bin
  export PYTHONPATH=~/lib/python
  ```

- You can use your own compiler-flags. For example:

  ```
  CFLAGS="-march=amdfam10 -O3 -msse4a -mfpmath=sse -pipe -fomit-frame-pointer" python setup.py
  ```

- If you have libraries in a not default path, please let it know.

  For example (e. g. OpenBSD or FreeBSD):

  ```
  export LIBRARY_PATH=$LIBRARY_PATH:/usr/local/lib
  export CPATH=$CPATH:/usr/local/include
  ```

- Because we are doing scientific computing (repeatable: conform to IEEE-754 standard; with the same result each time the computing is done — independent of the load of the machine) the default floating point arithmetic unit is set to sse by the use flag "-mfpmath=sse" in "setup.py".

  If you really wish to use 387 read the gcc man page to understand what you do. In principle if you calculate with 387 the results may be different every time! Caused by the simplex algorithm mostly used in ttrr through GLPK the basis of the result will mostly be the same. The final calculation of the objective function value is not very time consuming. Therefore the results could be the same.

- SciPy has missing dependencies in Gentoo Linux amd64 at the moment (2012-02-26) (see https://bugs.gentoo.org/show_bug.cgi?id=380597). You need sci-libs/clapack and sci-libs/lapack-atlas, too. For example:

  ```
  emerge sci-libs/clapack sci-libs/lapack-atlas
  eselect blas set atlas
  eselect cblas set atlas
  eselect lapack set atlas
  emerge sci-libs/scipy
  ```

- Fedora needs not only glpk but glpk-devel and gcc, too. gcc should come as a dependency of SciPy, but it does not.

- Fedora lacks multimedia software. The repository http://rpmfusion.org/ provides MJPEG, ffmpeg2theora and FFmpeg (and more), but not povray. It is possible to install it manually (see other hint).

- Since in debian the module argparse is not included in python, one needs the package python-argparse in addition.

- However, in debian png2yuv from MJPEG http://mjpeg.sourceforge.net/ is not in the package management system. But it can be installed as mjpegtools from: http://deb-multimedia.org

- If povray is not in the package management system (ubuntu 12.04, Fedora 16, Fedora 17, LMDE 201204, debian 6.0.5), it can be installed from http://www.povray.org/ http://www.povray.org/redirect/www.povray.org/ftp/pub/povray/Official/Linux/povlinux-3.6.tgz

  Unfortunately the install script does not work with a simple shell. Replacing sh with bash in the first line of the install script gives a working script in many distributions. (Sometimes /bin/sh is a link to /bin/bash and then it is working out of the box.)

- openSUSE 12.1 x86_64 behave in a different way from typical linux.

  The software SciPy and GLPK are not in the package management system; but are available from the repository http://download.opensuse.org/repositories/science/ (see http://www.scipy.org/Installing_SciPy/Linux#head-3dbbf9abe395e974d3ec911ae56863f4af70df9b).

  Not only glpk but also glpk-devel is needed.

  To install ttrr software to your $HOME in openSUSE the following steps are to perform:

  ```
  tar xzf ttrr-*.tar.*
  cd ttrr-*/
  python setup.py build_c_scripts
  python setup.py install --prefix=~
  ```

  Please make sure to have these pathes in $PATH and $PYTHONPATH, respectively. For openSUSE you need:

  ```
  export PATH=$PATH:~/bin
  export PYTHONPATH=~/lib/python2.7/site.packages
  ```

The software ffmpeg2theora and FFmpeg are in the repository: http://packman.links2linux.org/

10.5 tested and developed

Tested and developed with Gentoo Linux http://www.gentoo.org/ and the gentoo-packages

- dev-lang/python
- dev-python/numpy
- sci-libs/scipy
- sci-mathematics/glpk
- sci-visualization/gnuplot
- media-gfx/povray
- media-video/mjpegtools
- media-video/ffmpeg2theora
- media-video/ffmpeg
- media-libs/libtheora
- media-gfx/apngasm
- media-gfx/imagemagick

in the following versions/USE-flags:

- sci-mathematics/glpk-4.45 USE="doc examples gmp -mysql -odbc -static-libs"
- dev-lang/python-2.7.2-r3 USE="doc examples gdbm ipv6 ncurses readline sqlite ssl threads tk (wide-unicode) xml -berkdb -build -wininst"
- sci-visualization/gnuplot-4.4.4-r1 USE="X cairo doc emacs examples gd ggi latex lua plotutils readline thin-splines wxwidgets xemacs (-svga)"
- media-gfx/povray-3.7.0_rc3 USE="X openexr tiff -debug"
- media-video/mjpegtools-2.0.0-r1 USE="gtk mmx png -dga -dv -quicktime -sdl -sdlgfx -static-libs -v4l"
- media-video/ffmpeg2theora-0.28 USE="-debug -kate"
- dev-python/numpy-1.6.0 USE="doc lapack -test"
- sci-libs/scipy-0.9.0-r1 USE="doc -umfpack"
- media-libs/libtheora-1.1.1 USE="doc encode examples -static-libs"
- media-gfx/apngasm-2.5
- media-video/ffmpeg-0.7.8 USE="3dnow 3dnowext X aac alsa bzip2 doc encode hardcoded-tables jpeg2k mmx mmxext mp3 oss qt-faststart speex ssse3 theora threads truetype vorbis zlib (-altivec) -amr -avx -bindist (-celt) -cpudetection -custom-cflags -debug -dirac -faac -frei0r -gsm -ieee1394 -jack -network -pic -rtmp -schroedinger -sdl -static-libs -test -v4l -vaapi -vdpau -vpx -x264 -xvid" VIDEO_CARDS="-nvidia"
- media-gfx/imagemagick-6.7.5.3 USE="X bzip2 cxx djvu fontconfig graphviz gs jpeg jpeg2k lzma openexr openmp pango perl png raw svg tiff truetype wmf xml zlib -autotrace -corefonts -fftw -fpx -hdri -jbig -lcms -lqr -opencl -q32 (-q64) -q8 -static-libs -test -webp"

10.5.1 tested with

ttrr was successfully tested with:

- Gentoo Linux amd64 http://www.gentoo.org/ (with Python2 and Python3; see hints above)

- ubuntu 11.10 desktop i386 http://www.ubuntu.com/
- ubuntu 11.10 desktop amd64 http://www.ubuntu.com/
- ubuntu 12.04 desktop amd64 http://www.ubuntu.com/ (since povray is not in the package management system, it was manually installed; see hints above)
- ubuntu 13.04 desktop amd64 http://www.ubuntu.com/ (since povray is not in the package management system, it was manually installed)
- FreeBSD 9.0 i386 http://www.freebsd.org/ (see hints above)
- FreeBSD 9.0 amd64 http://www.freebsd.org/ (see hints above)
- OpenBSD 5.0 i386 http://www.openbsd.org/ (see hints above)
- OpenBSD 5.0 amd64 http://www.openbsd.org/ (see hints above)
- OpenBSD 5.1 amd64 http://www.openbsd.org/ (see hints above)
- Fedora 16 x86_64 Desktop http://fedoraproject.org/ (without povray; see hints above)
- Fedora 17 x86_64 Desktop http://fedoraproject.org/ (since povray is not in the package management system, it was manually installed; see hints above)
- LMDE 201204 with MATE 64-bit http://www.linuxmint.com/ (since povray is not in the package management system, it was manually installed; see hints above)
- debian 6.0.5 amd64 http://www.debian.org/ (since povray is not in the package management system, it was manually installed; see hints above)
- Arch Linux 2011.08.19 core x86_64 http://www.archlinux.org/ (with Python3)
- openSUSE 12.1 x86_64 http://www.opensuse.org/ (with required additional repositories; see hints above)

10.6 after install

You can check your installation with the examples in the examples directory:

```
./test
```

The results are compared with the reference results in "results.tar.bz2". You should view the images and pictures with your preferred software, too.

Under GLPK 4.52.1 you get different binary results compared to "results.tar.bz2". But these are only numerical rounding errors.

The most time (about 98 % cpu time) is spent on rendering by povray.

If the bash script "test" is not working for you, you can try single examples, e. g.:

```
python batch_2_load_cases_3_dim.py
python compareresults.py --compare_results -f *.bz2
ttrr_tools.py --png_gnuplot --png_pov --ogv --apng --mng --gif -f 2_load_cases_3_dim_tar.bz2 -xm
```

On a laptop (lenovo T410s, 4 intel-cores with 2.53 GHz, 4 GB RAM) it takes about 4 seconds for test_short and about 178 minutes (real time, 478 minutes user time) for test_long with povray V3.6.

On a desktop computer (8 amd-cores with 3.60 GHz, 16 GB RAM) it takes about 12 seconds for test_short and about 70 minutes (real time, 365 minutes user time) for test_long with povray V3.7.

CHAPTER

ELEVEN

LICENSE: TTRR - TRUSSTOPOLOGY-ROBUSTREDUNDANT

Author: Daniel Mohr.

Date: 2013-11-09 (last change).

License: GNU GENERAL PUBLIC LICENSE, Version 3, 29 June 2007.

CHAPTER

TWELVE

GPL

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for

software and other kinds of works.

The licenses for most software and other practical works are designed

to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not

price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you

these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether

gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains

that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run

modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.

States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and

modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of

works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this

License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work

in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based

on the Program.

To "propagate" a work means to do anything with it that, without

permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other

parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices"

to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work

for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official

standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other

than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all

the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users

can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that

same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of

copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not

convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under

the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological

measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid

circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you

receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey,

and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to

produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

> a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
>
> b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
>
> c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
>
> d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent

works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms

of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

> a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
>
> b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
>
> c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
>
> d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
>
> e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to

the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded

from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any

tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods,

procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or

specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a

requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided,

in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this

License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option

remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you

add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

> c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
>
> d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
>
> e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
>
> f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further

restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you

must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the

form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly

provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your

license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is

reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the

licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or

run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically

receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an

organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the

rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this

License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims

owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free

patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express

agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license,

and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or

arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within

the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting

any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or

otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have

permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of

the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the

Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future

versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different

permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY

APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING

WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided

above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest

possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest

to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short

notice like this when it starts in an interactive mode:

> <program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school,

if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program

into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

CHAPTER

THIRTEEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

BIBLIOGRAPHY

[Ben-TalElGhaouiNemirovski2009] Aharon Ben-Tal ; Laurent El Ghaoui ; Arkadi Nemirovski: Robust Optimization. Princeton University Press, 2009. http://press.princeton.edu/titles/9099.html

[BendsoeSigmund2004] Bendsoe, Martin P. ; Sigmund, Ole: Topology Optimization. Berlin, Heidelberg : Springer, 2004. ISBN 3-540-42992-1

[Marti2003] Kurt Marti: Stochastic optimization methods in optimal engineering design under stochastic uncertainty. In: ZAMM, 83 (12): 795–811, 2003. http://dx.doi.org/10.1002/zamm.200310072

[Michell1904] Michell, Anthony George M.: The limits of economy of material in frame structures. In: Phil. Mag. 8 (1904), Nr. 47, S. 589-597. http://dx.doi.org/10.1080/14786440409463229.

[Mohr2011] Mohr, Daniel P.: Redundante Topologieoptimierung. Neubiberg, Universitaet der Bundeswehr Muenchen, Fakultaet fuer Luft- und Raumfahrttechnik, Diss., Dezember 2011. http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:bvb:706-2664

[MohrSteinMatziesKnapek2012] Mohr, Daniel P. ; Stein, Ina ; Matzies, Thomas ; Knapek, Christina A.: Robust Topology Optimization of Truss with regard to Volume. In: arXiv - Mathematics, Optimization and Control (2012). http://arxiv.org/abs/1109.3782v2

[MohrSteinMatziesKnapek2013] Mohr, Daniel P. ; Stein, Ina ; Matzies, Thomas ; Knapek, Christina A.: Redundant robust topology optimization of truss. In: Optimization and Engineering (2013), 1-28. http://dx.doi.org/10.1007/s11081-013-9241-7

[PattersonGibsonKatz1987] Patterson, David A. ; Gibson, Garth A. ; Katz, Randy H.: A Case for Redundant Arrays of Inexpensive Disks (RAID). Version: Dec 1987. 1987 (UCB/CSD-87-391). — Forschungsbericht. — EECS Department, University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/1987/5853.html

[Przemieniecki1968] Janusz S. Przemieniecki: Theory of matrix structural analysis. McGraw-Hill, Inc., New York, 1968.

[Shannon1948] Shannon, Claude E.: A Mathematical Theory of Communication. Version: 1948. 1948 (1). — Forschungsbericht. — 379-423, 623-656 S. — The Bell System Technical Journal. http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html

PYTHON MODULE INDEX

t