6. Saxon Simulation Meeting

Mehrkritierielle Parameteroptimierung eines Thermoelektrischen Generators

Alexander Heghmanns

Prof. Dr.-Ing. Michael Beitelschmidt

Chemnitz, 01.04.2014

Inhalt

Einleitung

Rahmenprojekt

- **HiTEG:** "***Hi****gh Temperature* ***T****hermo* ***E****lectric* ***G****enerators*" (03X3548H)

Partner

- Material, Module: CoorsTek/ANCeram, Fraunhofer IKTS, O-Flexx
- Automobil: BMW, Faurecia, Institut für Automobiltechnik Dresden (TU-Dresden)
- Industriebrenner: NOXMAT
- Schienenfahrzeug: Bombardier Transportation, Institut für Festkörpermechanik (TU-Dresden)

GEFÖRDERT DURCH
Bundesministerium für Bildung und Forschung

Motivation

Steigende Energiekosten

Politisches und öffentliches Bewusstsein für Umweltschutz

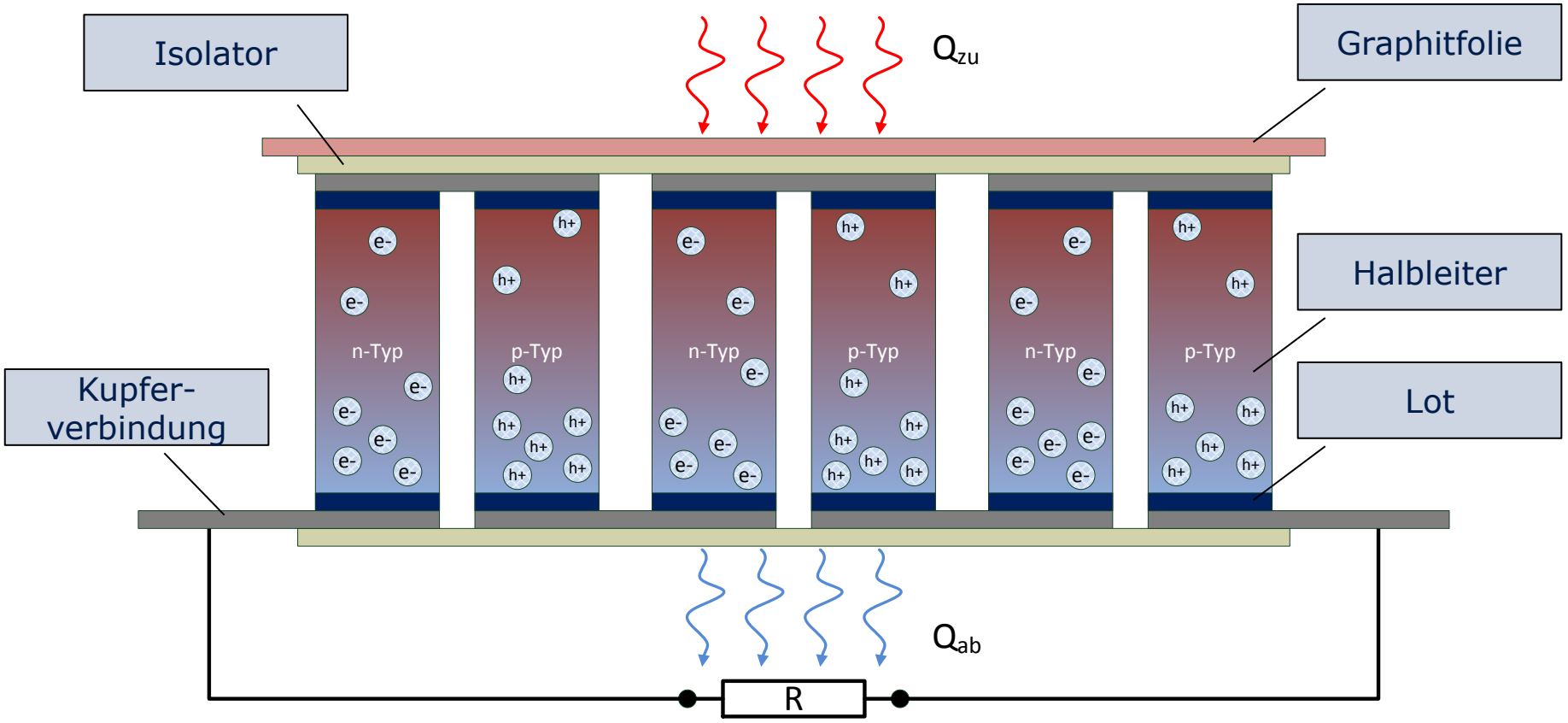
Abwärme in Industrie und mobilen Anwendungen

Lokomotive (Source: Bombardier)

LKW (Source: Wikipedia)

Reduktion des Energiebedarfs durch Abwärmenutzung mit Hilfe von thermoelektrischen Generatoren

Thermoelektrischer Generator



$$U_{Seebeck} = \alpha \cdot (T_H - T_C)$$

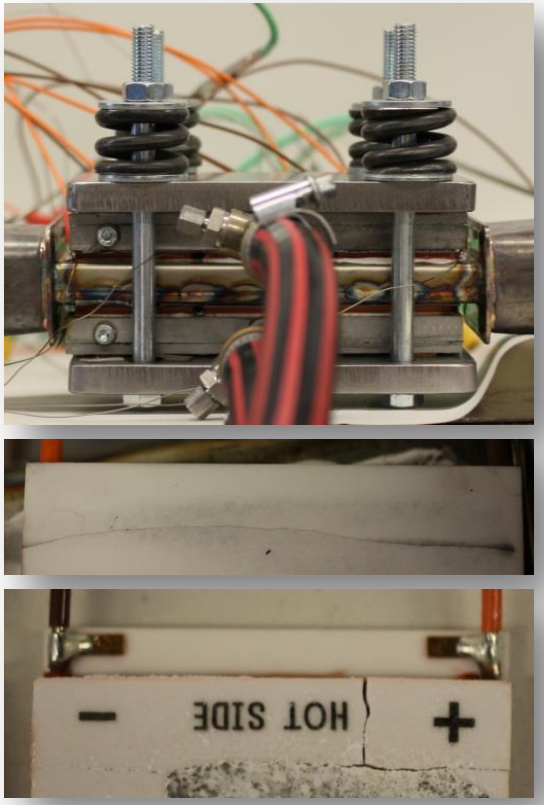
$$\dot{Q}_{Peltier} = -\alpha_{n,p} \cdot I \cdot T$$

$$\dot{q}_{Thomson} = -T \cdot J \cdot \frac{d\alpha}{dT}$$

Applikation von thermoelektrischen Generatoren (TEG)

- Aufgaben:
  - thermische Anbindung
  - Temperaturgradient muss möglichst über dem TE-Material anliegen
  - Hohe thermomechanische Spannungen können Bauteil zerstören
  - Wirtschaftlichkeit muss sichergestellt werden

- Möglicher Lösungsweg:
  - Optimierung des TEG hinsichtlich maximaler Leistung und minimaler mechanischer Belastung mit Hilfe der FEM



Gebrochenes TEG-Modul auf Zyklenprüfstand (Quelle: Faurecia)

Prinzipielles Vorgehen und Annahmen

- Modellanforderungen für Optimierung:
  - schnell lösbar
  - hinreichend genau
- Annahmen:
  - schwach gekoppeltes System ➔ thermisches, mechanisches und thermoelektrisches Problem getrennt voneinander lösbar
  - statische Vorgänge
  - linearelastische Berechnung für gewählte Zielfunktionen zulässig ➔ Überprüfung erforderlich

$$K_T \cdot T = q \qquad K_K(T) \cdot u = f(T) \qquad P_{el}(TE) = \sum_{i=1}^{n_{Halbleiter}} \mathrm{f}_{0D}(T_{max,i}, T_{min,i})$$

Berechnung der thermoelektrischen Ausgangsleistung

- Vereinfachter 0D-Ansatz bietet hinreichende Genauigkeit [JUNIOR, 2010; ROWE, 2006; SANDOZ, 2009]
- Bestimmung temperaturabhängiger Parameterbestimmung bei mittlerer Temperatur

$$T_M = \frac{T_H + T_C}{2} \qquad R_i(T_M) = \frac{h_L \cdot \left[\rho_{p,M} + \rho_{n,M}\right]}{A_L}$$

- Wärmestrom, elektrische Spannung und elektrischer Strom direkt berechenbar

$$u(T_M) = \frac{1}{2} \cdot (T_H - T_C) \cdot (\alpha_{p,M} - \alpha_{n,M})$$

$$i(T_M) = \frac{u}{R_{load} + R_i}$$

$$q_{H/C,p/n} = \mp\frac{1}{2} \cdot R_{el} \cdot i^2 \pm \alpha_{p/n,M} \cdot i \cdot T_H + \frac{\kappa_{p/n,M} \cdot A}{h} \cdot \Delta\mathrm{T}$$

- Elektrische Leistung unter Annahme optimaler Lastanpassung

$$P_{el}(T_M) = i^2 \cdot R_{load}$$

Zielfunktionen

- mechanische Zielfunktion:

  (Mechanischer Ausnutzungsgrad eines Testgesamtvolumens der sortierten, nicht fehlerbehafteten Spannungen, gewichtet mit dem Volumen der Elemente)

$$j_{mech,i} = \left(\frac{\sum_{e=0}^{n_i} \frac{\min\left(\sigma_{i,s,e}, \sigma_{i,max}\left(T_{s,e}\right)\right)}{\sigma_{i,max}\left(T_{s,e}\right)} \cdot V_{i,s,e} \cdot \delta}{\sum_{e=0}^{n_i} \left(V_{i,s,e} \cdot \delta\right)} \right) \rightarrow min$$

$$\sum_{e=0}^{n_i} \left(V_{i,s,e} \cdot \delta\right) \geq V_{i,min} : n_i \rightarrow min \qquad \delta = \begin{cases} 1 \; wenn \; \epsilon_{error,i,s,e} \leq \epsilon_{error,lim} \\ 0 \; sonst. \end{cases}$$

- elektrische Zielfunktion:

$$j_{elektr} = -\frac{P_{el}(T_M)}{V_{TEG}} \rightarrow min$$

Bewertungsfunktionen

- mechanische Bewertungsfunktion:
  (Gewichtete und invertierte Summe aller mechanischen Ausnutzungsgrade ➔ besonderer Einfluss der durch den Parametersatz beeinflussten Komponenten)

$$\xi_{mech,i} = 1 - j_{mech,i}$$

$$w_{mech,i} = \frac{std(j_{mech,i})}{\sum std(j_{mech})}$$

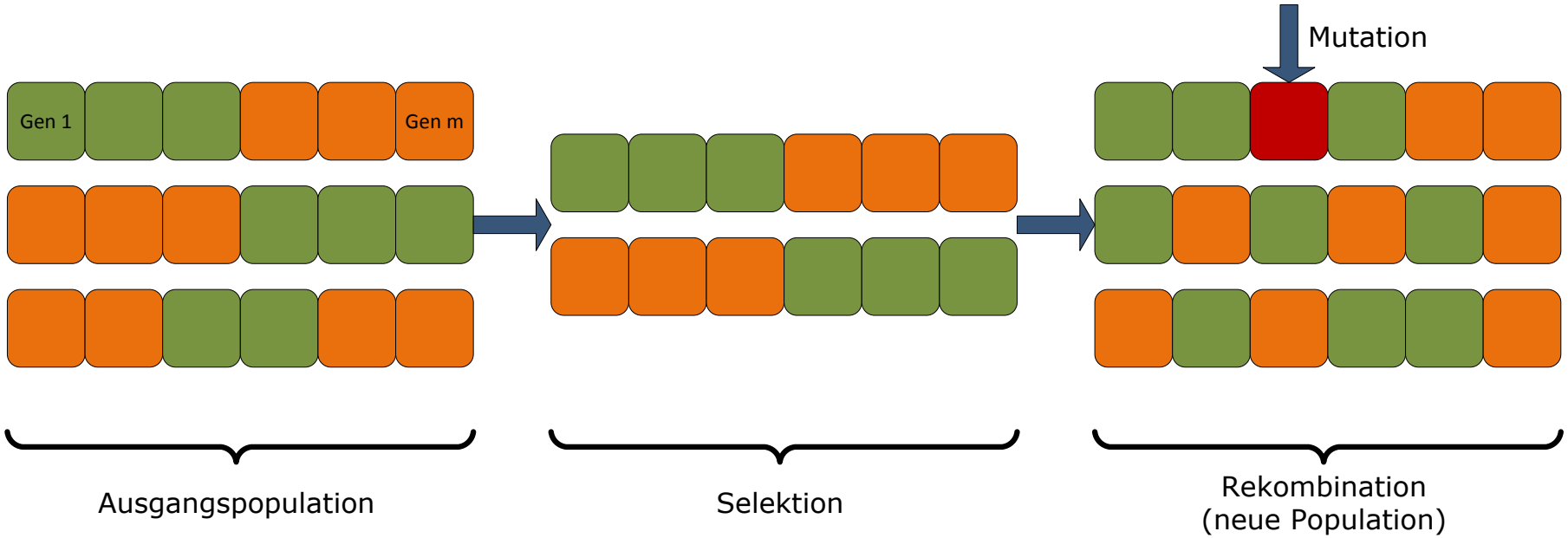
$$\Xi_{mech} = \sum \xi_{mech,i} \cdot w_{mech,i}$$

- thermische Bewertungsfunktion:

$$\xi_{elektr} = -j_{elektr}$$

Optimierungsverfahren

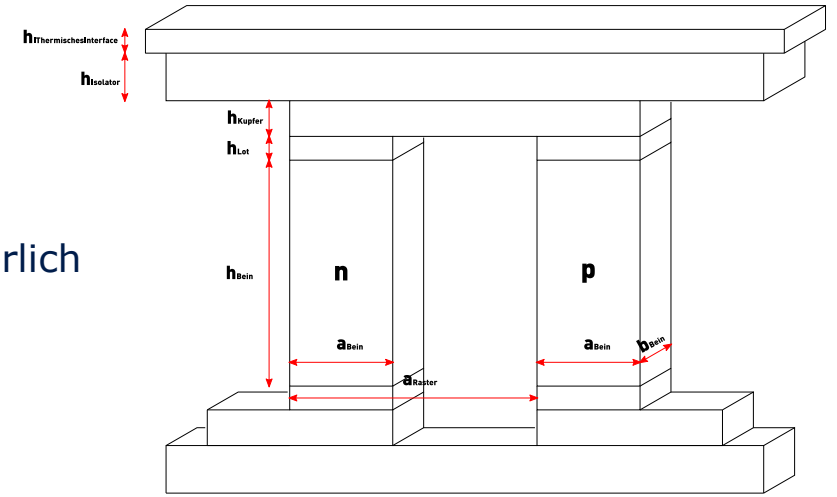
- Genetische mehrkriterielle Optimierung mit MATLAB GLOBAL OPTIMIZATION TOOLBOX
- FE-Berechnung in ANSYS WORKBENCH 15.0



Inhalt

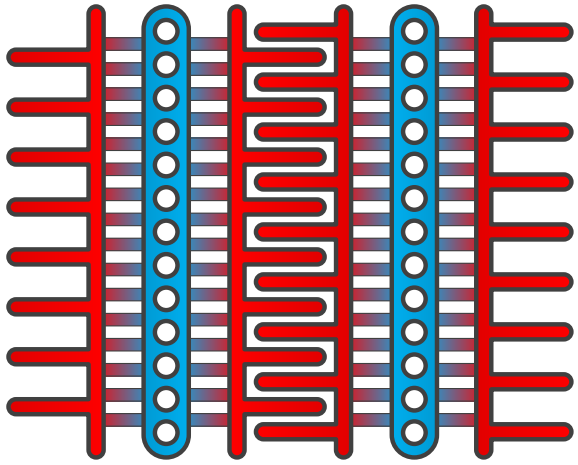
Parameter

- Schenkelhöhe, -breite und Anzahl
- Isolatorhöhe
- Schenkelanzahl ➔ Programmiertes Modell erforderlich



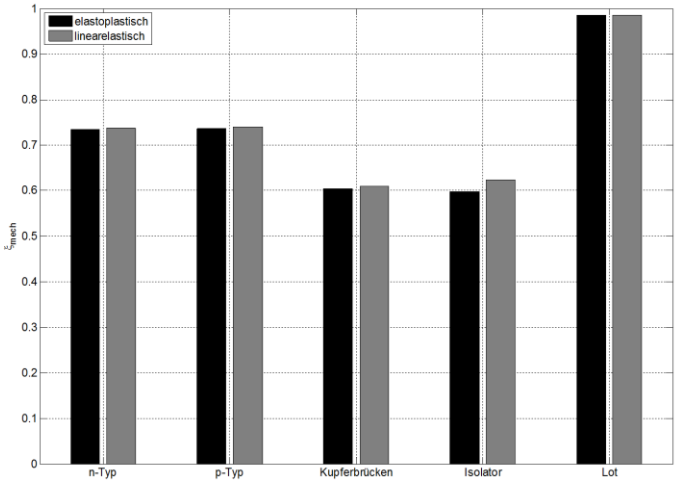
Randbedingungen

- Grundfläche des TEG ist konstant 30x30 mm²
- Anpressdruck für Graphitfolie: 1 MPa
- T_H = 220° C; T_K = 25° C (spannungsfreier Zustand)
- Translatorische Freiheitsgrade auf Grundfläche der kalten Seite gesperrt
- Elastische Lagerung mit Schubsteifigkeit von je 5 nebenstehenden TEG

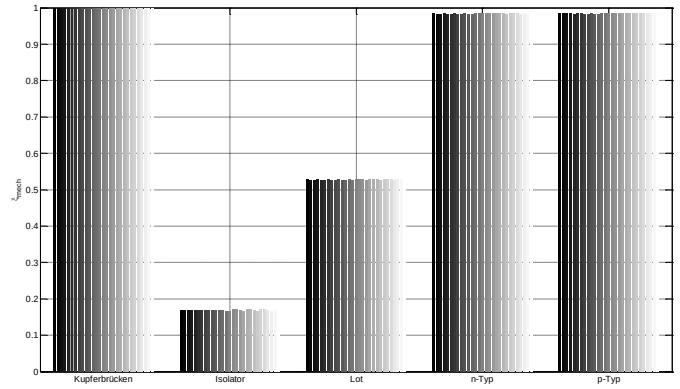


Prinzipielles Vorgehen und Annahmen

- Modellanforderungen für Optimierung:
  - schnell lösbar
  - hinreichend genau
- Annahmen:
  - schwach gekoppeltes System ➔ thermisches, mechanisches und thermoelektrisches Problem getrennt voneinander lösbar
  - statische Vorgänge
  - linearelastische Berechnung für gewählte Zielfunktionen zulässig ➔ Überprüfung erforderlich
  - Parameterstudie: Reibkontakt Graphitfolie nur einseitig erforderlich ➔ Wärmetauscherseite



Vergleich der mechanischen Bewertungsfunktionen für linearelastisches und elastoplastisches Materialverhalten



Vergleich der mechanischen Bewertungsfunktionen für verschiedene Reibkontaktmodellierungen an der Graphitfolie. Wenigstens eine Seite ist reibungsbehaftet.

Modellprogrammierung

- Nutzung der Scriptsprache JScript (entspricht weitestgehend JavaScript)
- Mechanical
  - „Workbench Macros“
  - Befehlsreferenz und Beispiele zum Teil verfügbar [ANSYS, 2012; ANS.NET, 2014]
- Design Modeler
  - Weitestgehend undokumentiert
  - Kaum Beispiele verfügbar
  - Befehle müssen im Debug-Modus oder in Quelldateien aufwändig recherchiert werden

Modelldebugging zur Befehlsrecherche im Design Modeler

- Verbindung der Ansys Workbench mit Debugger (z.B. MS Visual Studio) [SUTTON, 2011]
- Debugger wird bei Aufruf des Befehls *debugger* oder bei Script-Fehler aufgerufen
- Gezieltes Suchen nach Befehlen anhand des Funktionsnamens, Recherchieren der Funktionsparameter
- Grundsätzliche Objektstruktur (Auswahl)
  - ag.m ➔ Modellhandling (Analysetool, Featureauswahl, …)
  - ag.b ➔ Modellerstellung (Extrudieren, FPoint erstellen, Ebene wählen, …)
  - ag.c ➔ Sammlung von Konstanten
  - ag.gui ➔ Modellerstellung (Primitives: Quader, Zylinder, …), Schnittebenen, …)
  - ag.tree ➔ Modellbaum
  - ag.agApplet.Script ➔ nützliche Befehle zum Scripting

Beispiel: Funktion zur Erstellung eines Quaders im Design Modeler

```
/**
  * Erstellt einen Quader (Primitive Type = 2) in der XY-Ebene
  * X, Y, Z:                          Abmessungen
  * X_origin, Y_origin, Z_origin:     Ursprung
  * frozen:                           true/false (gefroren hinzufügen / Material hinzufügen)
  */
function createCuboid(X, Y, Z, X_origin, Y_origin, Z_origin, frozen, name)
{
    var myQuader = ag.gui.CreatePrimitive(2);
    myQuader.BaseX = X; myQuader.BaseY = Y; myQuader.BaseZ = Z;

    myQuader.DiagonalX = X; myQuader.DiagonalY = Y; myQuader.DiagonalZ = Z;
    myQuader.OriginX = X_origin; myQuader.OriginY = Y_origin; myQuader.OriginZ = Z_origin;
    myQuader.Name = name;

    if(frozen) { myQuader.Operation = 15; }
}
```

Script zum Aufruf der Funktion

```
// Funktionen einlesen
eval((new ActiveXObject("Scripting.FileSystemObject")).OpenTextFile("PATH_TO_FUNCTIONFILE", 1).ReadAll());
// Quader mit eigener Funktion erstellen
createCuboid(5, 5, 5, 0, 0, 0, false, "Quader_1");
// Modell erstellen
ag.b.Regen();
```

Modelldebugging zur Befehlsrecherche im Mechanical

- Gleiches Vorgehen wie im Design Modeler
- Grundsätzliche Objektstruktur (Auswahl)
  - DS.Tree ➔ DesignSpace Baumstruktur („TREE“)
  - TREE.FirstActiveBranch ➔ Erster Zweig (aktuelles Projekt) („BRANCH“)
  - („BRANCH“).Components ➔ Komponenten
  - („BRANCH“).Loads ➔ Lasten
  - („BRANCH“).ExternalLoads ➔ Externe Lasten
  - („BRANCH“).MeshControls ➔ Netzeinstellungen
  - DS.Script ➔ Nützliche Befehle und Scripte
  - SM ➔ SelectionManager
  - SM.PartMgr ➔ PartManager

Beispiel: Funktion zum Zuweisen eines Materials anhand der Bounding Box im Mechanical

```
/**
  * Exportiert alle Resultelements als ASCII-Dateien
  * ACHTUNG: Knotennummer und Koordinaten mit exportieren -> Extras::Optionen::Mechanical::Export
  * input:
  *     fileDir         ... Zielverzeichnis für Ergebnisdateien
  *     fileExt         ... Dateiendung (z.B. "txt")
  */
function exportAllResults(fileDir, fileExt)
{
    var n_res = DS.Tree.FirstActiveBranch.AnswerSet.Children.Count;
    for (i = 1; i <= n_res; i++) {
        try {
            var result      = DS.Tree.FirstActiveBranch.AnswerSet.Children(i);
            var resultName  = fileString(result.Name);
            var fileText    = result.CreateTabbedFile(fileDir + resultName + "." + fileExt);
            DS.Script.WriteTextInFile(fileDir + resultName + "." + fileExt, fileText);
        }
        catch(ex) {
            // wenn das Ausführen der Schleife nicht ging, war das Children kein Lösungsobjekt.
            // (z.B. das "Lösungsinformations"-Objekt)
        }
    }
}
```

Script zum Aufruf der Funktion

```
// Wenn das Script aus dem Projekt per Python aufgerufen wird, müssen diese beiden Objekte erzeugt werden.
var DS = WB.AppletList.Applet("DSApplet").App;
var SM = DS.SelectionManager;

// Funktionen einlesen
eval((new ActiveXObject("Scripting.FileSystemObject")).OpenTextFile("PATH_TO_FUNCTIONFILE", 1).ReadAll());

exportAllResults("PATH_TO_RESULT_DIR", ".txt")
```

Hinweis: Weitere Funktionsbeispiele für Design Modeler und Mechanical sind im Anhang.

Manueller Aufruf der Makros

- Design Modeler: Datei → Skript ausführen
- Mechanical: Extras → Makro ausführen

Automatisierter Aufruf der Makros

- Python-Script in Ansys Workbench: Datei → Skripterstellung → Skriptdatei ausführen

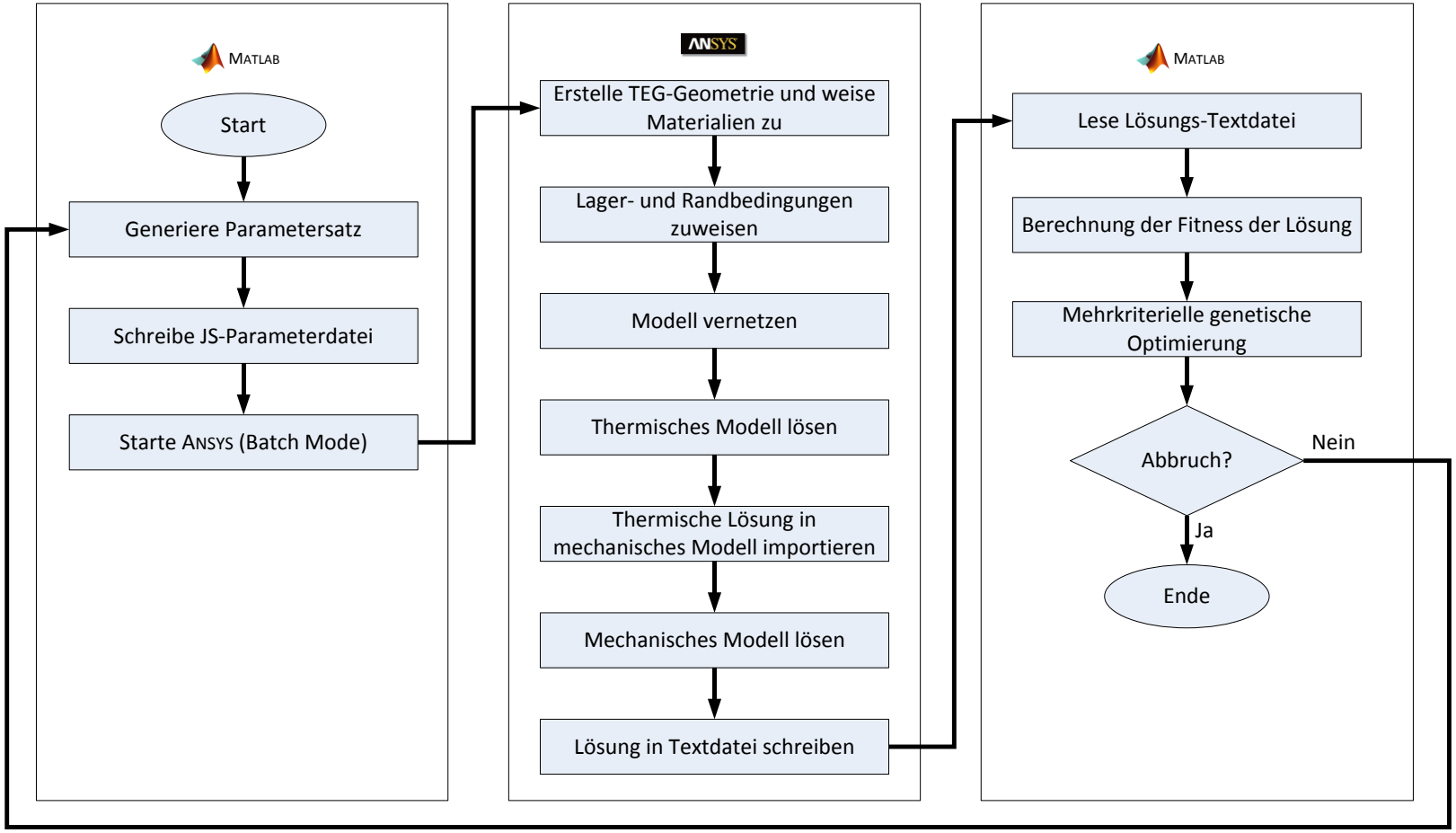
```
import os
Open(FilePath = "PATH_TO_MODEL") # Open the Workbench Project

# Component Geometry holen und öffnen
system1 = GetSystem(Name="Geom")
geometry1 = system1.GetContainer(ComponentName="Geometry")
geometry1.Edit()

# Sende JScript-Command to geometry1
script = open('PATH_TO_JSCRIPT_FILE', 'r')
geometry1.SendCommand(Command=script.read())
script.close()

geometry1.Exit()
```

Batchbefehl zum Start des Modells mit Python Script

```
"C:\Program Files\ANSYS Inc\v150\Framework\bin\Win64\RunWB2" -B -R PATH_TO_PYTHON_FILE
```

Inhalt

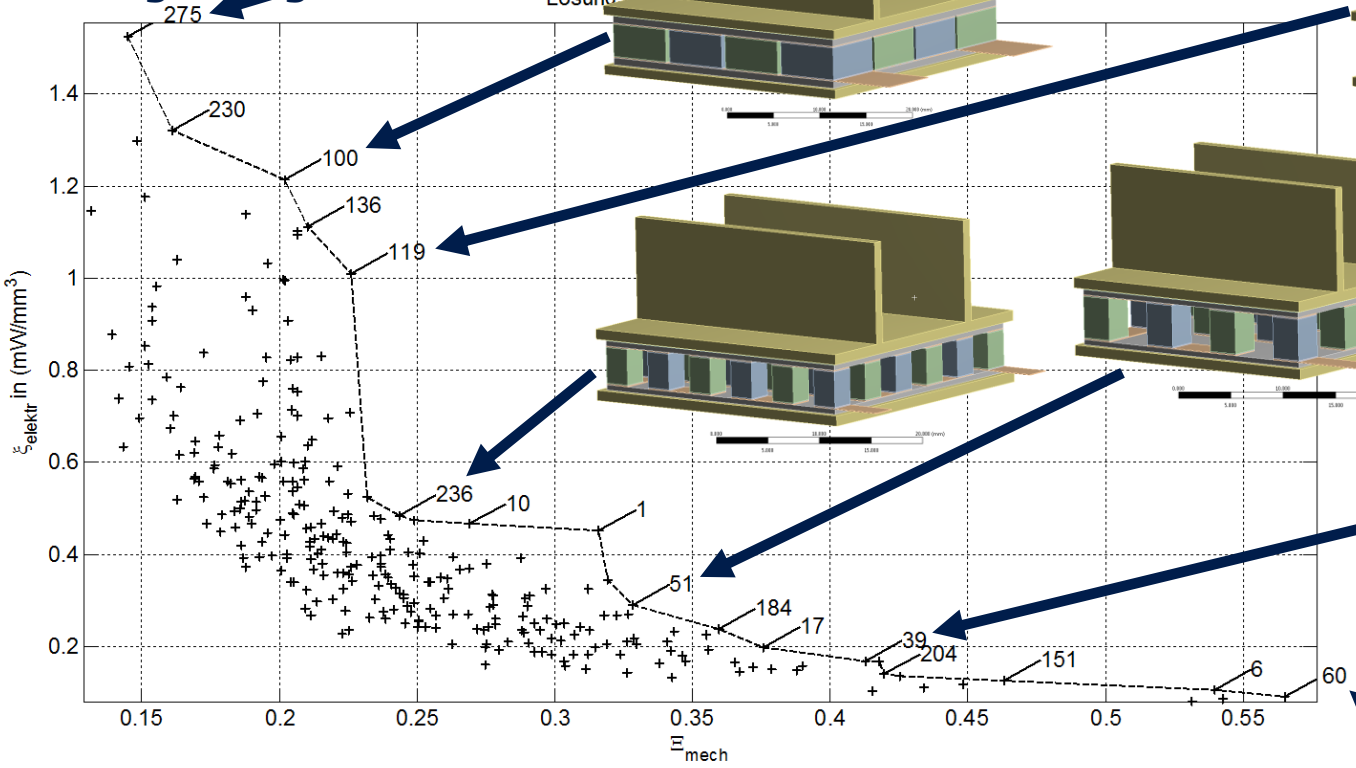
Prinzipielle Umsetzung

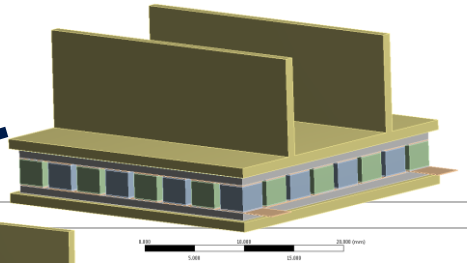


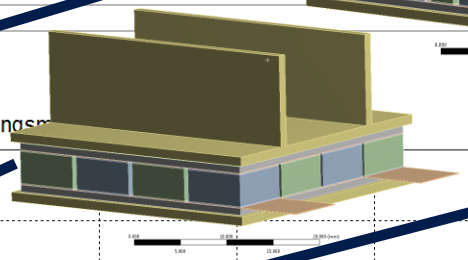
Inhalt

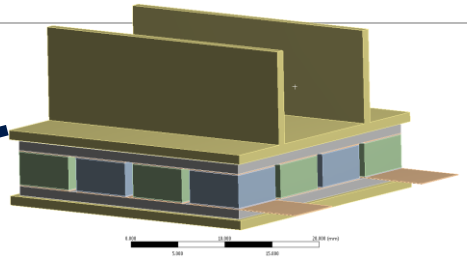
Ergebnisse

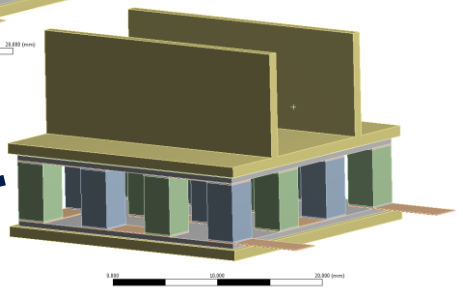
Lösungsmenge

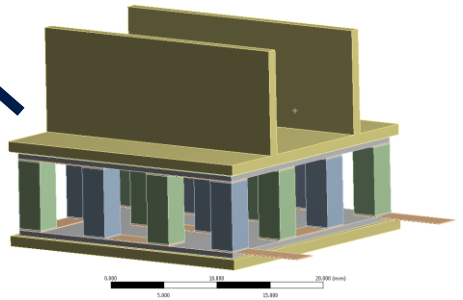


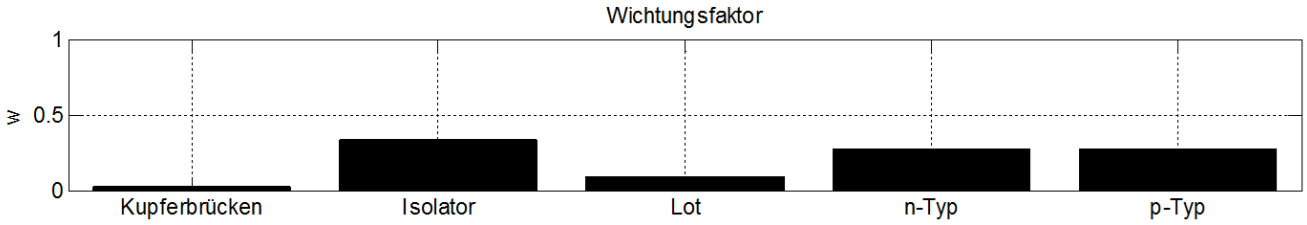












Inhalt

Zusammenfassung

- Thermoelektrische Generatoren (TEG) sind eine Möglichkeit zur Effizienzsteigerung von Verbrennungsprozessen
- Die Optimierung des TEG ist zwingend notwendig um eine maximale Ausgangsleistung bei ausreichender mechanischer Festigkeit zu erreichen
  - Modellvereinfachungen müssen dazu getroffen werden
  - Mehrkriterielle genetische Optimierungsalgorithmen sind geeignet
- Ansys Workbench bietet mit Hilfe der Script-Schnittstellen die Möglichkeit Modelle frei zu programmieren und eine Parameteroptimierung durchzuführen

Ausblick

- Detaillierte Analyse einiger gewählter Lösungen auf der Paretofront
- Weitere Arbeiten an neuen TEG-Designs zur Erhöhung der Wirtschaftlichkeit
- Test des Wirkprinzips an Demonstrator

Inhalt

Literaturquellen

[ANSYS, 2012] ANSYS Mechanical User Guide, Release 14.5, 2012

[JUNIOR, 2010] JUNIOR, C.: Analyse thermoelektrischer Module und Gesamtsysteme, Technische Universität, Braunschweig, Diss., 2010

[ROWE, 2006] ROWE, D. M.: General Principles and Basic Considerations. In: Thermoelectrics Handbook. Taylor & Francis, 2006

[SANDOZ, 2009] SANDOZ-ROSADO, E. J.: Investigation and Development of Advanced Models of Thermoelectric Generators for Power Generation Applications, Rochester Institute of Technology, Diss., 2009

Internetquellen

[ANS.NET, 2014] ANSYS.NET: Macro Library, Workbench JavaScript Macros, 2014. http://ansys.net/?mycat=search&mytype=Macros&mycategory=Workbench, letzter Aufruf: 02.03.2014

[SUTTON, 2011] SUTTON, M.: ANSYS Mechanical Scripting: HOWTO Part 4, 2011. http://www.padtinc.com/blog/the-focus/ansys-mechanical-scripting-howto-part-4, letzter Aufruf: 02.03.2014

Inhalt

Design Modeler: Mit einem Quader schneiden

```
/**
  * Erstellt einen Schnittebene mit einem Quader (Primitive Type = 2, Operation = 13) in der XY-Ebene
  * X, Y, Z:                          Abmessungen
  * X_origin, Y_origin, Z_origin:    Ursprung
  */
function sliceWithCuboid(X, Y, Z, X_origin, Y_origin, Z_origin, name)
{
    var myQuader = ag.gui.CreatePrimitive(2);

    myQuader.BaseX = X; myQuader.BaseY = Y; myQuader.BaseZ = Z;
    myQuader.DiagonalX = X; myQuader.DiagonalY = Y; myQuader.DiagonalZ = Z;
    myQuader.OriginX = X_origin; myQuader.OriginY = Y_origin; myQuader.OriginZ = Z_origin;
    myQuader.Name = name;

    myQuader.Operation = 13;
}
```

Design Modeler: Einen Zylinder erstellen

```
/**
  * Erstellt einen Zylinder (Primitive Type = 4) in der XY-Ebene
  * R, L:                          Abmessungen
  * X_origin, Y_origin, Z_origin:  Ursprung
  * frozen:                        true/false (gefroren hinzufügen / Materian hinzufügen)
  */
function createCylinder(L, R, X_origin, Y_origin, Z_origin, frozen, name)
{
    var myCylinder = ag.gui.CreatePrimitive(4);

    myCylinder.InnerRadius = R; myCylinder.AxisZ = L;
    myCylinder.OriginX = X_origin; myCylinder.OriginY = Y_origin; myCylinder.OriginZ = Z_origin;
    myCylinder.Name = name;

    if(frozen)
    {
        myCylinder.Operation = 15;
    }
}
```

Design Modeler: FPoints aus Koordinatendatei erzeugen

```
/**
  * Erstellt eine Punkteliste "FPoint" aus einer Koordinatendatei
  *
  */
function createFPointsFromFile(file, name)
{
    var myFPoint = ag.b.FPoint(ag.c.FPointConstruction, ag.c.FPointCoordinateFile);
    myFPoint.Name = name;
    myFPoint.CoordinateFile = file;

    ag.b.Regen();

    return myFPoint;
}
```

Design Modeler: Oberfläche aus geschlossenen Linien erzeugen

```
/**
  * Durchsucht den Strukturbaum nach searchName und erstellt
  * aus den Kanten des gefundenen Objekts Oberflächen.
  */
function createSurfaceFromClosedLines(searchName, surfaceName)
{
    // Strukturbaum holen
    var nodes = ag.tree.Nodes;
    var surfaceCounter = 1;
    // Hinten im Baum beginnen, die ersten Objekte (Ebenen, etc.) weg lassen
    for(i1 = 1; i1 <= nodes.Count; i1++)
    {
        var myNode = nodes(i1);
        if(myNode._ObjectDefault.search(searchName) != -1) {
            var mySurface = ag.b.SurfFromLines();
            mySurface.name = surfaceName +"_"+ convertIntToStringWithLeadingZeros(surfaceCounter, 2);
            var myLine = ag.m.GetFeatureFromLabel(myNode._ObjectDefault);
            var i = 1;
            var myEdge;
            while(myEdge = myLine.getEdge(i)) {
                mySurface.Add3dedge(myEdge);
                i++;
            }
            surfaceCounter++;
        }
    }
    ag.b.Regen();
}
```

Mechanical: Alle Lasten und Randbedingungen im Modell löschen

```
 /**
   * Entfernt sämtliche Lasten im Modell.
   */
function deleteAllLoads()
{
    // Anzahl der Lasten
    var countLoads = DS.Tree.FirstActiveBranch.Loads.Count;
    var loadsIDs = new Array();

    // IDs der Lasten auslesen
    for(i = 1; i <= countLoads; i++) {
        var myLoad = DS.Tree.FirstActiveBranch.Loads(i);
        loadsIDs.push(myLoad.ID);
    }

    // Objekte der Lasten anhand der ID löschen
    for(i = 0; i < loadsIDs.length; i++) {
        DS.Tree.DeleteObject(loadsIDs[i]);
    }

    // Strukturbaum neu aufbauen
    DS.Script.fillTree();
}
```

Mechanical: Findet Flächen mit einem bestimmten Flächeninhalt

```
/**
 * Sucht Flächen mit dem Flächeninhalt area innerhalb einer festgelegten Tolleranz.
 */
function findFacesByArea(area, tolerance)
{
    var faces = new Array();
    SM.Clear();

    DS.Graphics.EntityFilter = DS.Script.id_FaceFilter;
    SM.SelectAll();

    for (var i = 1; i <= SM.SelectedCount; i++)
    {
        // Oberfläche holen
        var partId = SM.SelectedPartID(i); var faceId = SM.SelectedEntityTopoID(i);
        var part = SM.PartMgr.PartById(partId); var brep = part.BRep;
        var face = brep.Cell(faceId);
        var entityId = SM.SelectedEntityRefIDAsInt(i);

        if(typeof(face.Area) == "undefined") { continue; }

        if(area*(1-tolerance) < face.Area && area*(1+tolerance) > face.Area) {
            faces.push(new Array(partId, faceId, entityId));
        }
    }
    SM.Clear();

    return faces;
}
```

Mechanical: Oberflächentemperatur mit findFacesByArea-Funktion erstellen (Auszug)

```
// alle Lasten entfernen
deleteAllLoads();

// Selektiere Flächen mit dem Flächeninhalt 100 mm^2 innerhalb einer Toleranz von 5%
var faces = findFacesByArea(100, 0.05);
SM.Clear();

// selektiere die erste Fläche und erstelle eine Oberflächentemperatur
SM.ForceSelect(faces[0], faces[1]);
var myLoadTop = DS.Tree.FirstActiveBranch.Environment.AddLoad(SM, DS.Script.id_SurfaceTemperature);

// Temperatur von 220° C einstellen. Die Variable muss ein String sein!
myLoadTop.Magnitude = "220";

// Strukturbaum neu aufbauen
DS.Script.fillTree();
```

Dipl.-Ing. Alexander Heghmanns

- Tel.: 0351 463 36628
- Email : jan_alexander.heghmanns@tu-dresden.de

Prof. Dr.-Ing. Michael Beitelschmidt

- Tel.: 0351 463 37970
- Email: michael.beitelschmidt@tu-dresden.de

»Wissen schafft Brücken.«