A Novel Algorithm for the Determination of Walker Damage in Loaded Disc Springs

Geilen, Max Benedikt; Klein, Marcus; Oechsner, Matthias

(2020)

materials

MDPI

Article

A Novel Algorithm for the Determination of Walker Damage in Loaded Disc Springs

Max Benedikt Geilen *, Marcus Klein and Matthias Oechsner

Centre for Engineering Materials (MPA-IfW); Technical University of Darmstadt; Grafenstraße 2, 64283 Darmstadt, Germany; m.klein@mpa-ifw.tu-darmstadt.de (M.K.); oechsner@mpa-ifw.tu-darmstadt.de (M.O.)

* Correspondence: geilen@mpa-ifw.tu-darmstadt.de; Tel.: +49-6151-16-25867

Received: 3 March 2020; Accepted: 30 March 2020; Published: 3 April 2020

Abstract: In this paper, a novel algorithm for the determination of Walker damage in loaded disc springs is presented. The algorithm takes a 3D-scan of a disc spring, measured residual stresses, material parameters, and spring loads as inputs. It outputs a distribution of Walker damage over the surface area of the input disc spring. As the algorithm allows a fully automated determination of the Walker damage, it can be used by disc spring manufacturers to reduce the working time spent on this task by specialized engineers significantly. Compared to spreadsheet applications using analytical formulas and finite element models using idealized geometry, this approach offers a superior description of the stress states in disc springs.

Keywords: disc springs; fatigue; FEA; automation; real geometry; Abaqus Scripting

1. Introduction

The fatigue behavior of disc springs can be described by models ranging widely in complexity. Simple models like the one described in [1,2] can be implemented using spreadsheet applications, e.g., Microsoft Excel. As models become more sophisticated, the expense of building and evaluating the model rises. With the latest step in creating more complex models, the introduction of scanned geometries, superposed residual stresses, and the Walker damage parameter, the need for a novel algorithm for the determination of Walker damage in loaded disc springs has arisen.

The algorithm described in this paper is implemented in the Spring_stack Python module, which can also be used for other applications. For more information on the simulation of single disc springs without a 3D-scanned geometry or with multiple springs in one assembly, see [3,4].

The algorithm is the first published algorithm to build and evaluate a finite element model from a 3D-scanned geometry without user interaction. It allows the user to obtain an understanding of the influence of geometric deviations that are present in a batch of disc springs on the lifetime of individual disc springs under cyclic loading. This would be impossible without the algorithm because manually conducting a finite element simulation for each disc spring is prohibitively expensive.

Introductions to the mathematical description of disc springs and to the Walker damage parameter are given in Section 2. In Section 3, an algorithm used to describe Walker damage at the surface of a 3D-scanned disc spring is presented. Its implementation is described in Section 4. An example application is presented in Section 5. We recommend that readers without programming experience read Section 5 before Section 4.

2. Related Research

2.1. Mechanical Behavior of Disc Springs

The first mention of conical disc springs with a rectangular cross-section in the literature was BELLEVILLE's patent in 1861 [5–7]. The first formulas to compute the characteristic of disc springs, as well as the stresses present in disc springs under load were published by ALMEN and LASZLO in 1936. These formulas are still in use in today's standards [1,2], supplemented by a friction formulation published by CURTI and MONTANINI [8]. There is a variety of other approaches for the analytical assessment of disc springs, each bringing its own advantages, such as improved accuracy [9–19], improved simplicity [20], applicability for different geometries [21–28], consideration of new material laws [29–31], offering a new concept for friction [32], or allowing the computation of resonance frequencies [33].

Analytical assessment of disc springs allows a direct understanding of the mechanical nature of disc springs and even the direct identification of links between geometric features and stresses. It also offers a sufficiently precise description of disc springs for applications like fatigue design according to EN 16983 and EN 16984 [1,2] at low computational costs. They can also be automated easily, which makes these approaches attractive for optimization algorithms like [34,35] and for analytical models of systems containing multiple components like [36,37].

Since the 1980s [38], disc springs have also been assessed using finite element analysis (FEA). While FEA is expensive in licensing, computational cost, and training, it allows models to be adapted to new, similar problems quickly. The first FEA models were used to exceed the accuracy of the analytical approaches available at that time [39,40]. FEA is still in use for the verification of new analytical models. FEA has extensively been used to describe residual stresses and changes in the characteristics created by plastic deformation and creep effects [41–46]. It has also been used to describe disc springs with complex geometries [47–51] or complex load cases [52], as well as for disc springs made from new materials [48,53–57] and to describe the behavior of assemblies containing disc springs [58–60].

Today, 2D models of disc springs solve quickly. An automated tool for the processing of 2D simulations of disc springs was implemented as early as 2000 [61]. FEA has been included in the curricula of most engineering programs. Therefore, graduates with basic FEA skills are available to companies. Disc springs have been numerically simulated by free finite element software [62]. These effects have led to the literature being split roughly in half between analytical formulas and FEA. The authors believe this distribution to roughly stay the same in the future because both approaches offer unique advantages and because of a legacy effect. The legacy effect is created by fatigue tests being evaluated using a certain method and the raw data like material laws and spring geometries not being documented. Furthermore, engineers are more experienced in the design of springs using analytical formulas, and this experience can only partially be used for the design of springs using FEA.

2.2. The Walker Damage Parameter

The Walker damage parameter [63] helps to compare the fatigue behavior of materials under different mean stresses. Compared with other approaches to the characterization of mean stress effects in steels [64–66], the Walker damage parameter shows a superior lifetime prediction [67]. In this paper, the stress-based approach is described. For the strain-based approach, see [68].

The Walker damage parameter P_{Walker} is computed from the maximum stress σ_{max}, the stress amplitude σ_{amp}, and the Walker exponent γ:

$$P_{\text{Walker}} = \sigma_{\text{max}}^{1-\gamma} \sigma_{\text{amp}}^{\gamma} \tag{1}$$

The Walker exponent γ is a material parameter that can be identified by fitting fatigue-curves with different mean stresses or R-ratios. A high Walker exponent implies a low sensitivity to mean

stress effects, while a low Walker exponent implies a high sensitivity. If the Walker exponent γ is fixed to 0.5, the stress Walker approach is equivalent to the stress Smith–Watson–Topper [66] approach.

The Walker equation requires σ_{max} and σ_{amp} to be scalar values. At the surface of the disc spring, the stress state is two-dimensional; therefore, an equivalent stress must be computed. Here, the von Mises equivalent stress is used.

For proportional loads in Quadrant 1, the computation of the maximum stress σ_{max} and the stress amplitude σ_{amp} from the minimum and maximum principal stresses $\sigma_{I,max}$, $\sigma_{II,max}$, $\sigma_{I,min}$, and $\sigma_{II,min}$ is obvious.

$$\sigma_{min} = \sqrt{\frac{\sigma_{I,min}^2 + (\sigma_{I,min} - \sigma_{II,min})^2 + \sigma_{II,min}^2}{2}} \tag{2}$$

$$\sigma_{max} = \sqrt{\frac{\sigma_{I,max}^2 + (\sigma_{I,max} - \sigma_{II,max})^2 + \sigma_{II,max}^2}{2}} \tag{3}$$

$$\sigma_{amp} = 0.5 \cdot (\sigma_{max} - \sigma_{min}) \tag{4}$$

The stress state in disc springs however is non-proportional, especially in shot peened specimens. An example why the formulas given for the proportional case may give inconsistent results for the non-proportional case is depicted in Figure 1. Given the stress states σ_a, σ_b, and σ_c in Quadrant 1, the stress amplitude between σ_a and σ_c is equal to the stress amplitude between σ_b and σ_c if the von Mises equivalent stress in σ_a is equal to the von Mises equivalent stress in σ_b. The same applies for mean stresses. In reality, the damage inflicted by cycling between σ_b and σ_c is smaller than the damage inflicted by cycling between σ_a and σ_c.

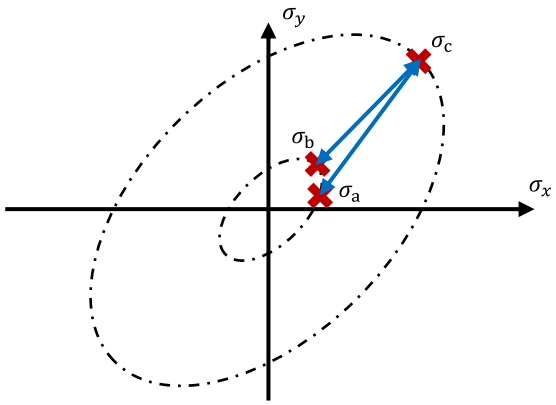

Figure 1. Planar stress with the von Mises equivalent (zero shear).

To avoid the described misrepresentation of damage inflicted, the modified Manson–McKnight method [69] is used. Instead of computing two scalar stress states σ_{max} and σ_{min} and deducing a scalar amplitude and a scalar mean stress, a two-dimensional stress amplitude and a two-dimensional mean stress are deduced from a two-dimensional maximum and minimum stress state:

$$\sigma_{I,amp} = 0.5 \cdot (\sigma_{I,max} - \sigma_{I,min}) \tag{5}$$

$$\sigma_{II,amp} = 0.5 \cdot (\sigma_{II,max} - \sigma_{II,min}) \tag{6}$$

$$\sigma_{I,mean} = 0.5 \cdot (\sigma_{I,max} + \sigma_{I,min}) \tag{7}$$

$$\sigma_{II,mean} = 0.5 \cdot (\sigma_{II,max} + \sigma_{II,min}) \tag{8}$$

These pairs of stress components are converted into von Mises equivalent stresses:

$$\sigma_{\text{amp}} = \sqrt{\frac{\sigma_{\text{I,amp}}^2 + \left(\sigma_{\text{I,amp}} - \sigma_{\text{II,amp}}\right)^2 + \sigma_{\text{II,amp}}^2}{2}} \tag{9}$$

$$\sigma_{\text{mean}} = \frac{\sigma_{\text{I,mean}} - \sigma_{\text{III,mean}}}{\sigma_{\text{I,mean}} + \sigma_{\text{III,mean}}} \cdot \sqrt{\frac{\sigma_{\text{I,mean}}^2 + \left(\sigma_{\text{I,mean}} - \sigma_{\text{II,mean}}\right)^2 + \sigma_{\text{II,mean}}^2}{2}} \tag{10}$$

The maximum stress σ_{max} is defined with a lower bound to avoid complex numbers as Walker damage parameters.

$$\sigma_{\text{max}} = \max\left(\sigma_{\text{mean}} + \sigma_{\text{amp}}, 0\right) \tag{11}$$

3. The Algorithm

3.1. General Concept

The algorithm starts with a surface mesh (given as an .STL file) of a disc spring placed randomly in space. After aligning the disc spring approximately symmetrically around the y-axis, it imports the geometry information into a finite element application. In our implementation of the algorithm, the finite element application was used. Through the Abaqus Scripting interface [70,71], it builds a model around the geometry data, using additional inputs like loads and friction coefficients provided by the user. It generates an output database file using the Abaqus/Standard Solver. Utilizing Abaqus/Viewer functionalities via the Scripting interface, a field of residual stresses is generated from input data. Aggregated minimum and maximum stress fields are computed by superposing the computed load stress field and the residual stress field. A Walker damage parameter field is computed from the minimum and maximum stress fields. The Walker damage parameter at the surface of the disc spring is computed and exported as tabular data.

3.2. Architecture

The software architecture used is called a pipeline in programming terms and resembles a production line: an instance of the class Spring_stack is passed through a series of different processing stations. The processing stations gradually transform the instance from input data to a solved finite element model with post-processed output fields and from there to easily readable tabular data and a graphical representation thereof. In programming terms, these processing stations are called methods. The flow of data is depicted in Figure 2.

3.3. User-provided Inputs

To conduct the inquiry described above, the algorithm requires several inputs. These inputs are provided by the user. They are comprised of an .STL file containing the triangle surface mesh of a scanned disc spring and one .JSON file for the description of the simulated physical situation and the numerical configuration of the simulation, respectively.

The .JSON file describing the simulated physical situation contains an array of parts defining the geometry, the Walker exponent γ, Young's modulus, the Poisson number, and a volumetric mass density (for numerical stability) for each. The geometry of scanned disc springs is given by means of a link to an .STL file. Plasticity may be defined for disc springs in the present implementation. However, this is not in the scope of this paper. Furthermore, the friction coefficients between springs stacked in parallel and springs stacked in series, as well as between springs and the plates and the pillar are committed. The axial load is committed as an array of paths, giving the displacements of one of the plates at the end of each time step. Measured residual stresses in the radial and tangential direction are committed for two points with different coordinates. These two points should be selected carefully by

the user and ideally describe a linear model representing more measurements. Furthermore, a target value for the number of triangles of the coarsened triangle surface mesh is committed.

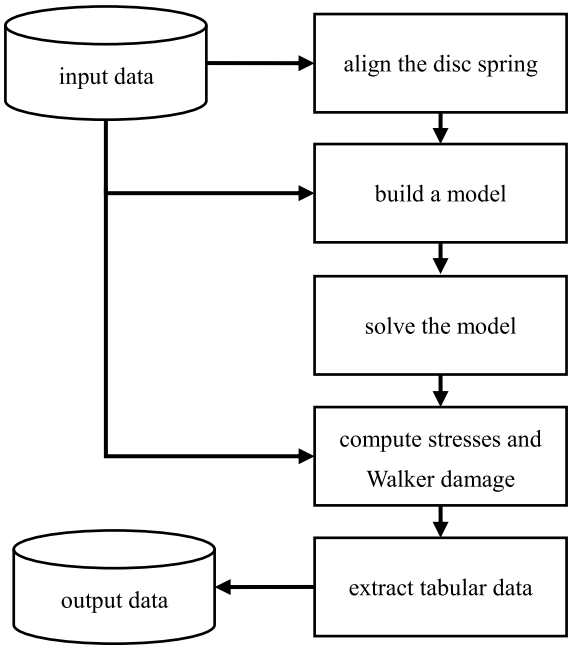

Figure 2. Simplified flowchart of the algorithm.

The .JSON file describing the numeric inputs contains several naming definitions, element types, solver options, contact and friction formulations, computing resource allowances, and flags determining whether stresses etc. are to be written to input files. Different modeling conventions for plates and pillars may be used in the current implementation. However, this is outside the scope of this paper.

4. Implementation

4.1. Aligning the Disc Spring

In order to build a model around a part, the alignment of the part in space must be known. For example, to apply axial loads, the axial direction of the loaded part must be known. Because it makes the rest of the algorithm much easier, the input disc spring is preprocessed so it always has the same orientation. By convention, this position is in the point of origin, closely axially symmetric to the y-axis.

In the alignment process, the mesh $\boldsymbol{\mathcal{M}}_{\text{start}}$ is processed as a set of vertices $\boldsymbol{V}_{\text{start}} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_v\}$ and a set of triangles $\boldsymbol{e} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_e\}$ connecting these vertices. The vertices $\boldsymbol{v}_{i,\text{start}} = (x_{i,\text{start}}, y_{i,\text{start}}, z_{i,\text{start}})$ are defined in a Cartesian coordinate system (x, y, z). The point set is aligned by a translation $\boldsymbol{t}\left(t_x, t_y, t_z, \boldsymbol{V}\right)$ along and a rotation $\boldsymbol{A}\left(\theta_x, \theta_y, \theta_z\right)$ around the principal axes.

$$\boldsymbol{V}_{\text{aligned}}\left(\boldsymbol{A}, \boldsymbol{t}, \boldsymbol{V}_{\text{start}}\right) = \boldsymbol{A}\left(\theta_x, \theta_y, \theta_z\right) \cdot \boldsymbol{t}\left(t_x, t_y, t_z, \boldsymbol{V}_{\text{start}}\right) \tag{12}$$

The edges are defined as connectors between vertices; therefore, a geometric transformation of any mesh $\boldsymbol{\mathcal{M}}$ is fully described by a geometric transformation of its vertices $\boldsymbol{V}$. The resulting mesh $\boldsymbol{\mathcal{M}}_{\text{aligned}}$ is defined by the vertices $\boldsymbol{V}_{\text{aligned}}$ and the edges $\boldsymbol{e}$.

A feasible combination of translation $\boldsymbol{t}$ and rotation $\boldsymbol{A}$ is found by solving an optimization problem.

$$\min_{\boldsymbol{\theta}_{x,y,z}, \boldsymbol{t}_{x,y,z}} f\left(\boldsymbol{V}_{\text{aligned}}\left(\boldsymbol{A}, \boldsymbol{t}, \boldsymbol{V}_{\text{start}}\right)\right) \tag{13}$$

To compute the objective function f, the transformed point cloud $\boldsymbol{V}_{\text{aligned}}$ is projected into the xy-plane, producing $\boldsymbol{V}_{xy} = \{\boldsymbol{v}_{xy,1}, \ldots, \boldsymbol{v}_{xy,v}\}$ by rotation around the y-axis. $\boldsymbol{V}_{xy}$ is computed using the coordinates of $\boldsymbol{V}_{\text{aligned}}$:

$$\boldsymbol{v}_{xy,i} = \begin{pmatrix} \sqrt{x_{i,\text{aligned}}^2 + y_{i,\text{aligned}}^2} \\ z_{i,\text{aligned}} \end{pmatrix} \tag{14}$$

The objective function f is defined as the area A of the convex hull of $\boldsymbol{V}_{xy}$.

$$f = A\left(Conv\left(\boldsymbol{V}_{xy}\right)\right) \tag{15}$$

The convex hull is computed using the Quickhull algorithm [72,73]. The objective function f is invariant to rotations around the y-axis. Therefore, θ_y is fixed to zero. To further simplify the optimization problem, it is assumed that the centroid of the aligned mesh $c\left(\boldsymbol{\mathcal{M}}_{\text{aligned}}\right)$ lies in the point of origin. Although this assumption is applicable only for perfectly symmetric spring geometries, we assumed that for springs possessing minor deviations regarding their symmetry, the assumption holds true, too.

Based on this assumption, $\boldsymbol{t}$ is defined as translating the volumetric centroid c of the mesh $\boldsymbol{\mathcal{M}}_{\text{start}}$ into the point of origin.

$$c\left(\boldsymbol{\mathcal{M}}_{\text{translated}}\right) = (0,0,0) \tag{16}$$

The computational cost for finding $\boldsymbol{t}$ is low because the volumetric centroid of any closed mesh $\boldsymbol{\mathcal{M}}$ can be computed inexpensively.

In the following rotation $\boldsymbol{A}$, the mesh $\boldsymbol{\mathcal{M}}_{\text{translated}}$ is only rotated around the principal axes. As the volumetric centroid lies on all three principal axes, it is invariant under said rotation $\boldsymbol{A}$.

$$c\left(\boldsymbol{\mathcal{M}}_{\text{aligned}}\right) = c\left(\boldsymbol{\mathcal{M}}_{\text{translated}}\right) = (0,0,0) \tag{17}$$

By fixing the translation additionally to the rotation around the y-axis, the dimensionality of the optimization problem is reduced from six to two.

$$\min_{\boldsymbol{\theta}_{x,z}} f\left(\boldsymbol{V}_{\text{aligned}}\left(\boldsymbol{A}, \boldsymbol{t}, \boldsymbol{V}_{\text{start}}\right)\right) \tag{18}$$

The parameters θ_x and θ_z are calculated using the L-BFGS-B method [73–75].

The resulting mesh $\boldsymbol{\mathcal{M}}_{\text{aligned}}$ may be aligned upside down. This being the case, it can simply be rotated by 180 degrees around the x-axis. To further ease the following steps, it is translated along the y-axis, so its lowest point is in the xz-plane.

$$\min\left(y_{i,\text{postprocessed}}\right) = 0 \tag{19}$$

The postprocessed, aligned mesh $\boldsymbol{\mathcal{M}}_{\text{postprocessed}}$ is exported as an Abaqus input file (.inp).

4.2. Building the Finite Element Model

The creation of the model is structured as a pipeline inside the pipeline depicted in Figure 2. The sub-pipeline is depicted in Figure 3. The allotment of tasks to processing stations (methods) along the pipeline is based on the allotment of functionalities to modules in the Abaqus/CAE environment. Each of the processing stations is customized according to input data. The following paragraphs describe the individual processing stations.

For the simulation of a single disc spring, four components are required: the disc spring itself, a guide pin, and two flat plates. The geometry of the disc spring is read from the input file generated from the 3D-scan data. The guide pin is defined as an idealized cylinder with the same height as the disc spring, and the plates are defined as idealized planes.

The plate and the guide pin part instances are defined as analytic rigid surfaces and therefore do not need to be meshed. The part instance representing the disc spring is imported as a triangle surface mesh and is converted into a volume tetrahedron mesh by the free mesher implemented in Abaqus/CAE. Mesh size cannot be controlled actively, but only by changing the mesh size of the imported surface mesh. The surface mesh size is adjusted by exporting very fine surface meshes from the 3D-scanning software GOM Scan and increasing the mesh size using the quadric edge collapse decimation algorithm [76]. In this application, quadric edge collapse is especially suitable because it reduces the mesh size at the flat surfaces, where the mesh may be more coarse, and keeps it nearly constant at the edges, where a fine mesh is needed.

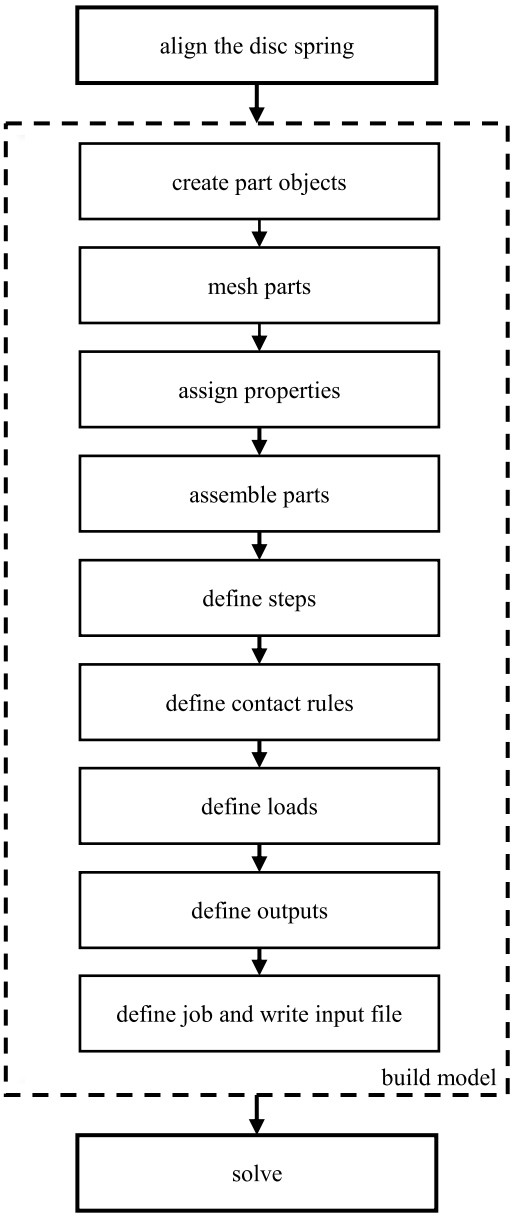

Figure 3. Flowchart of the sub-pipeline 'build model'.

In the approach described in this paper, the finite element analysis utilizes a purely elastic material law. Residual stresses are incorporated by superposition of the measured stresses committed by the user. This is implemented by building two axisymmetric field outputs, one for $\sigma_{\text{residual,tan}}$ and one for $\sigma_{\text{residual,rad}}$, in Abaqus/CAE. These output fields will later be used for the computation of $\sigma_{\text{I,min}}$, $\sigma_{\text{I,max}}$, $\sigma_{\text{II,min}}$, and $\sigma_{\text{II,max}}$ according to Equations (24) to (27).

To introduce inertia and improve convergence, a volumetric mass density is defined for the disc spring. Because the spring was aligned beforehand, the assembly is straightforward. All four parts are joined in an assembly with a single coordinate system. The lower plate, the disc spring, and the guide pin are already in place. The upper plate is positioned in its place by a translation along the y-axis.

The lower plate and the guide pin are fixed in space, and loads are applied to the upper plate. To improve convergence, a small gravitational load is defined. No other loads are applied directly to the disc. Forces are transmitted via contacts. Surface to surface contacts are defined between the disc spring and each of the other part instances.

The Abaqus Scripting interface includes an option to customize output requests. This option is used in the algorithm to request stress and coordinate outputs. However, coordinates cannot be requested at integration points by means of the Scripting interface. Coordinate data in the integration points are necessary to compute residual stresses in the integration points.

A job instance is created to generate an input file. The input file is manipulated to create an output request for coordinate information in the integration points, and a second job instance pointing to the new input file is created. This way, a job is created that is identical to the first job except that it includes an output request for coordinate information in the integration points. The second job is converted into a system of partial differential equations and solved using Abaqus/Standard.

4.3. Computing Stresses and Walker Damage Parameters

After solving the system of partial differential equations for the displacements of the nodes in the model, Abaqus/Standard computes stresses and coordinates in the integration points at different points in step-time, which refer to the minimum and the maximum load. In this section, the computation of Walker damage parameters at the surface of the disc spring between Edge II and Edge III (see Figure 4) is described. Here, we may assume a plane stress state with a dominating tension component:

$$\sigma_{\text{load,II,min}} = 0 \vee \sigma_{\text{load,III,min}} = 0 \tag{20}$$

$$\sigma_{\text{load,II,max}} = 0 \vee \sigma_{\text{load,III,max}} = 0 \tag{21}$$

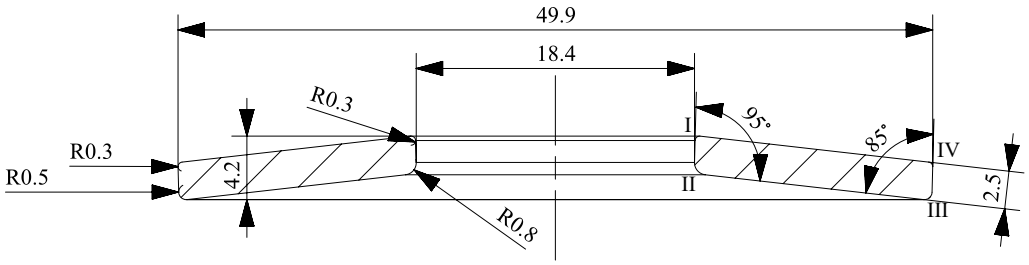

Figure 4. Idealized geometry of the spring under investigation.

For integration points close to the surface, this is approximately true. The assumption becomes more realistic with a finer mesh and is true with an infinitesimal mesh size. The minimal and maximal stress components $\sigma'_{\text{load,II,min}}$ and $\sigma'_{\text{load,II,max}}$ are redefined:

$$\sigma'_{\text{load,II,min}} = \sigma_{\text{load,II,min}} + \sigma_{\text{load,III,min}} \tag{22}$$

$$\sigma'_{\text{load,II,max}} = \sigma_{\text{load,II,max}} + \sigma_{\text{load,III,max}} \tag{23}$$

In the surface of disc springs, $\sigma_{\text{load,I}}$ points in the tangential direction and $\sigma'_{\text{load,II}}$ points in the radial direction. Residual stresses are measured in the tangential direction ($\sigma_{\text{residual,tan}}$) and in the

radial direction ($\sigma_{\text{residual,rad}}$). The input stresses for the computation of the Walker damage parameters are defined as:

$$\sigma_{\text{I,max}} = \sigma_{\text{load,I,max}} + \sigma_{\text{residual,tan}} \tag{24}$$

$$\sigma_{\text{II,max}} = \sigma'_{\text{load,II,max}} + \sigma_{\text{residual,rad}} \tag{25}$$

$$\sigma_{\text{I,min}} = \sigma_{\text{load,I,min}} + \sigma_{\text{residual,tan}} \tag{26}$$

$$\sigma_{\text{II,min}} = \sigma'_{\text{load,II,min}} + \sigma_{\text{residual,rad}} \tag{27}$$

Walker damage parameters are computed according to Equations (1) to (11). For each surface triangle, a surface area A_i and a Walker damage parameter $P_{\text{Walker},i}$ are computed. The surface Walker damage parameter is computed by averaging the Walker damage parameters of the neighboring nodes. These data pairs are saved as the set S.

4.4. Extracting Tabular Data

The set S is too rich in information to be captured holistically without extraction and/or condensation. This is done by transforming the information into tabular data. Therefore, it is condensed into human readable tabular data. Each element of the set is defined as a Dirac delta function:

$$f_i\left(P_{\text{Walker}}\right) = A_i \cdot \delta\left(P_{\text{Walker}} - P_{\text{Walker},i}\right) \tag{28}$$

The accumulated surface area A_{acc} assigned to a given Walker damage parameter is computed as:

$$A_{\text{acc}}\left(P_{\text{Walker}}\right) = \int_{P_{\text{Walker}}}^{\infty} \sum_i f_i\left(P_{\text{Walker}}\right) \mathrm{d}P_{\text{Walker}} \tag{29}$$

It describes the size of the surface area of the disc spring with a Walker damage equal to or greater than the given P_{Walker}. A graphical representation [77] of the accumulated surface area is created.

5. Example Application

In this section, an example application of the algorithm presented in Section 3 is given. A single disc spring was modelled. The surface mesh of the disc spring under investigation is presented in Figure 5. The mesh was obtained using the commercially available 3D-scanning device GOM ATOS and the software GOM Scan. The number of surface triangles was already reduced for the displayed mesh, from 253,904 to 50,152. Convergence studies with different loads showed this mesh to be a good compromise between computational cost (about seven hours of CPU time on a i7-9800X and 16 GB of RAM vs. about 110 h of CPU time and 115 GB of RAM for the 253,904 surface triangle model) and accuracy (no significant bias of the Walker damage-surface area plot) [4].

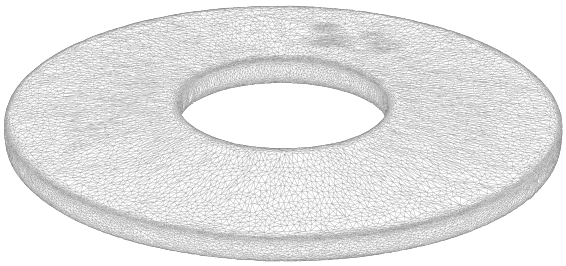

Figure 5. 3D-scanned, coarsened surface mesh of the disc spring under investigation.

Additionally, an idealized geometry derived from the scanned data is presented in Figure 4. Edges I to IV are labelled for the reader's orientation. A major difference between both geometries is that the surfaces between Edges I and IV, as well as II and III of the idealized geometry are straight, while those

of the 3D-scanned disc spring are curved. This is also visible in Figure 6. Of course, the 3D-scanned geometry also was not perfectly symmetric.

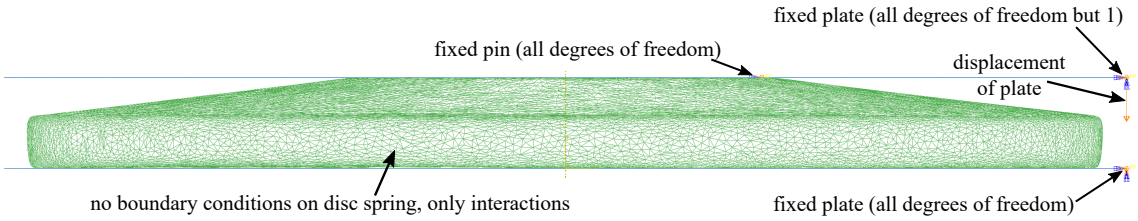

Figure 6. Boundary conditions defined for the finite element model.

In Figure 7, the geometry according to the standard [1] is presented. It differs from the idealized geometry in having sharp edges and all angles between faces being 90°. This simplified geometry is usually utilized for the analysis of disc springs, regardless of whether analytical formulas or finite element models are used.

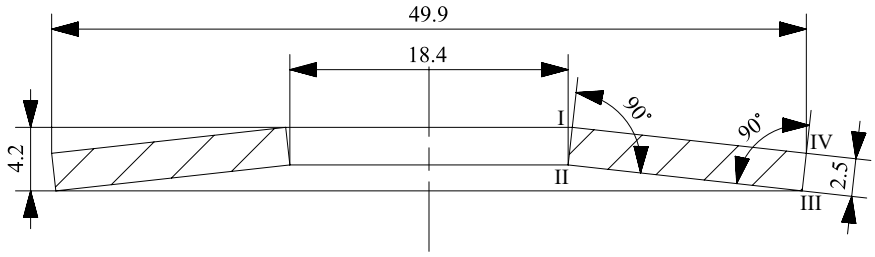

Figure 7. Geometry of the spring under investigation according to EN 16983 [1].

The triangle surface mesh was aligned and imported into Abaqus/CAE. Afterwards, it was converted into a tetrahedron volume mesh using the `generateMesh` method included in Abaqus Scripting [71]. The resulting linear tetrahedron C3D4 mesh was converted into a quadratic C3D10 mesh. Modified quadratic tetrahedrons C3D10M offer improved contact behaviour [78]; however, due to poor mesh quality, C3D10 tetrahedrons performed better in our experience. The elements were assigned a Young's modulus of 206,000 MPa, a Poisson ratio of 0.3, and a volumetric mass density of 8.05 g/mm^3.

The volumetrically meshed disc spring was incorporated into an assembly. The load cycle was implemented in four steps. To obtain good convergence behavior, the steps were defined as dynamic steps. The implicit solver Abaqus/Standard was used. The purpose of Step 1 was to apply the lower load. Step 2 was to make sure there was no dynamic influence on the computed stresses. Step 3 was to apply the higher load. Step 4 was, again, implemented to eliminate dynamic influences. Step times for Steps 1 to 4 were 10 s, 1 s, 10 s. and 1 s. The boundary conditions were applied to the upper plate as displacements in the axial direction at a reference point; see Figure 6. The prescribed displacements were 0.425 mm for Steps 1 and 2 and 1.19 mm for Steps 3 and 4. All other degrees of freedom of the reference points were fixed to zero. For the lower plate, all degrees of freedom were fixed to zero.

All contact formulations used in this model were defined as surface-to-surface contacts with a finite sliding penalty formulation and a friction coefficient of 0.01.

A job was created and committed to the solver Abaqus/Standard. The resulting output database was loaded. The resulting highest tensile load stresses in a cross-section are displayed in Figure 8. The upper half of the disc spring was loaded compressively. This is why disc springs in general break between Edges II and III.

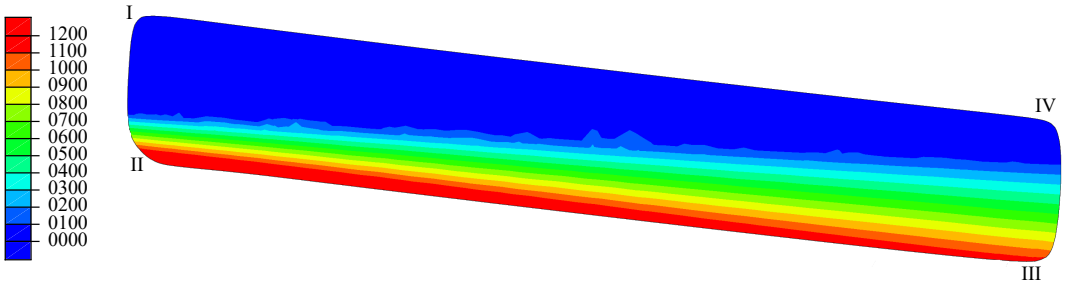

Figure 8. Computed compressive (deep blue, without detailed scale) and computed highest tensile (colored scale) load stresses in the fully loaded cross-section in MPa.

Two field outputs representing residual stresses in the tangential and radial direction were generated based on user input and the initial coordinates of the integration points (which were requested as outputs earlier). Since the measured residual stresses for the spring under investigation are confidential, the used inputs values were not the result of a measurement. They were however realistic for disc springs. The residual stresses between Edges II and III were approximated by a linear function, ignoring non-symmetric effects. The computed residual stresses were obviously wrong anywhere else. This does not matter here because the Walker damage parameter is a measure used to predict fracture. Fracture is initiated by cracks, which normally initiate from the surface between Edges II and III [79]. Cracks originating from between Edges I and IV are usually caused by too low preloading forces. The bias in the Walker damage parameter introduced outside the surface between Edges II and III is non-conservative. Therefore a false positive for fracture in this area can be ruled out. The computed output fields between Edges II and III are displayed in Figure 9. The significantly higher residual stresses in the tangential direction compared to the radial direction are normal in disc springs because disc springs are overloaded in production to prevent plastic deformation in use, to increase lifetime, and to decrease creep effects [80–85].

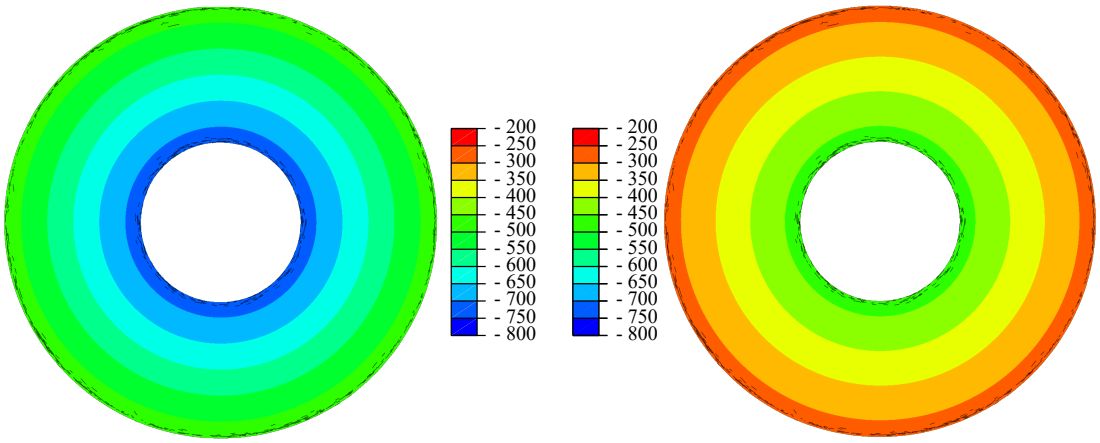

Figure 9. Residual stresses $\sigma_{\text{residual,tan}}$ and $\sigma_{\text{residual,rad}}$ in the tangential (left) and radial (right) direction in MPa.

The output fields resulting from the finite element simulation describing stresses after Step 2 and the output fields describing the residual stresses were added up to compute output fields describing $\sigma_{\text{I,max}}$ and $\sigma_{\text{II,max}}$. The calculation followed Equations (24) and (25). Field outputs describing $\sigma_{\text{I,min}}$ and $\sigma_{\text{II,min}}$ were computed following Equations (26) and (27).

Based on these, field outputs describing the minimum and maximum equivalent stress, σ_{min} and σ_{max}, were computed according to Equations (2) and (3); see Figure 10. Especially in the minimum equivalent von Mises stress visualization, the contact line between the disc spring and the lower plate can be identified by a circle of locations with high compressive stresses.

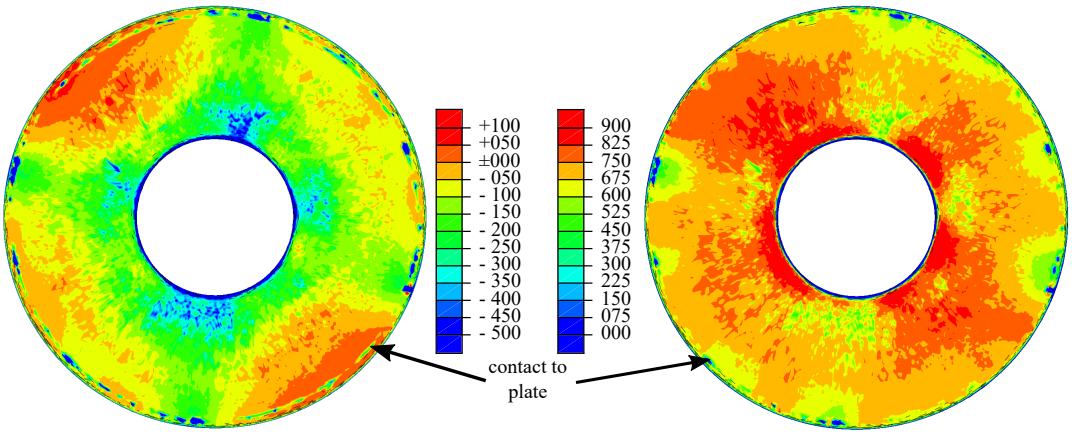

Figure 10. Minimum and maximum equivalent von Mises stress σ_{min} (left) and σ_{max} (right) in MPa.

These output fields were used to compute the output fields representing the stress amplitude σ_{amp} according to Equation (4) and finally the Walker damage parameter P_{Walker} according to Equation (1); see Figure 11. The Walker exponent $\gamma = 0.5$ was used, which makes the Walker damage parameter equivalent to the Smith–Watson–Topper damage parameter. Compared to the other fields, the stress amplitude field was very smooth. The reason for this is that the elastic deformation of the spring partially compensated for the small asymmetries that were present. For higher load increments, the additional elastic stresses were therefore distributed more homogeneously.

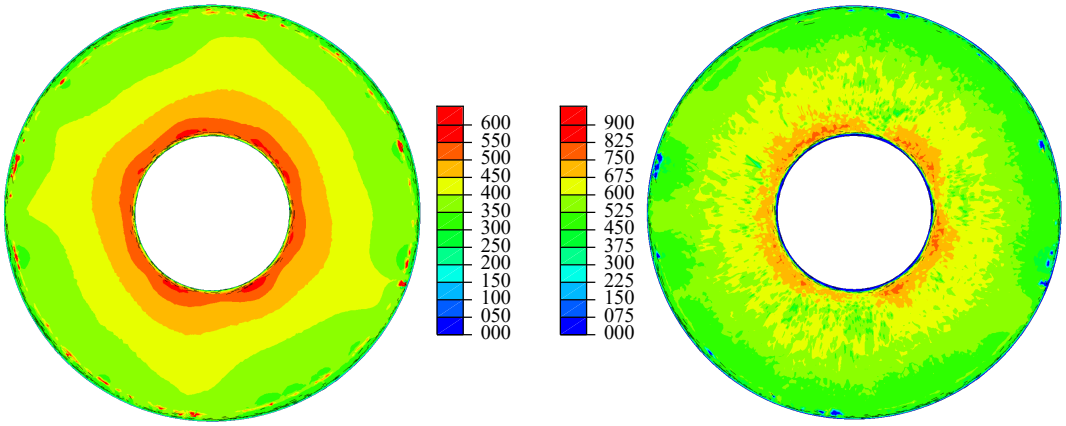

Figure 11. Stress amplitude σ_{amp} (left) and Walker damage P_{Walker} (right) in MPa.

The algorithm created a list of all surface triangles and computed an average Walker damage parameter $P_{\text{Walker},i}$, as well as a surface area A_i for each triangle. The graph of the accumulated surface area over the Walker damage parameter was created; see Figure 12. From this particular graph, the user can for example extract the surface area where the Walker damage parameter is over 750 MPa, which is 4.9 mm^2. About half of the surface had a Walker damage parameter of zero because stresses there were purely compressive; see Figure 12. This surface corresponds to the dark blue parts of the surface in Figure 8. As can be seen on the graph on the right, the function starts to jump at high stresses. This is because Walker damage parameters were averaged over surface triangles. The part of the graph at very high stresses exists purely because of numerical singularities and therefore does not correspond to the fatigue behaviour of the disc spring.

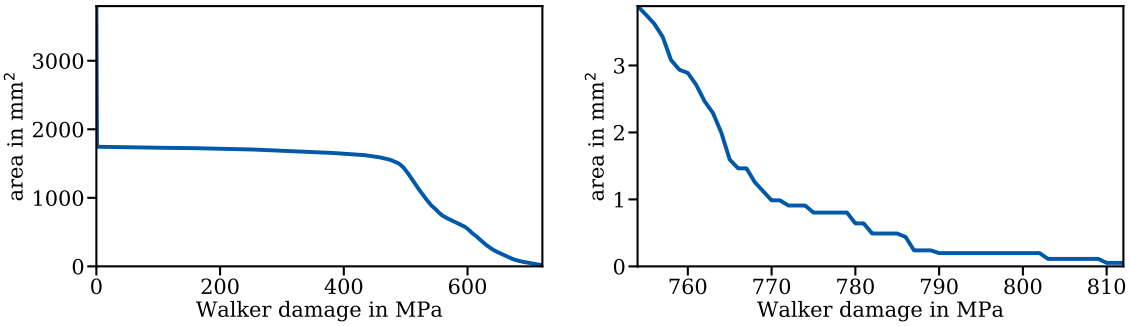

Figure 12. Accumulated surface area of the disc spring over Walker damage.

The Walker damage parameter is not directly accessible through experiments. To evaluate the quality of the finite element model, a characteristic derived from a similar model (only boundary conditions were changed) was compared to a characteristic obtained in an experiment; see Figure 13. As is customary for disc springs, the deflection was normalized over the deflection at which the disc spring lies flat on the ground. They agree well; especially, the correct representation of the progressive behaviour at the very start of the experiment has only been achieved by models directly implementing 3D-scanned geometry. To our knowledge, all published models directly implementing 3D-scanned geometry have been created using the algorithm presented in Section 3. The numerical characteristic is somewhat stiffer than the experimental one. This may be due to a misrepresentation of Young's modulus, which was set to the normative default of 206,000 MPa; however, tensile tests on specimens from the same batch of material and a similar heat treatment did not show a sufficient deviation in Young's modulus to use a lower value.

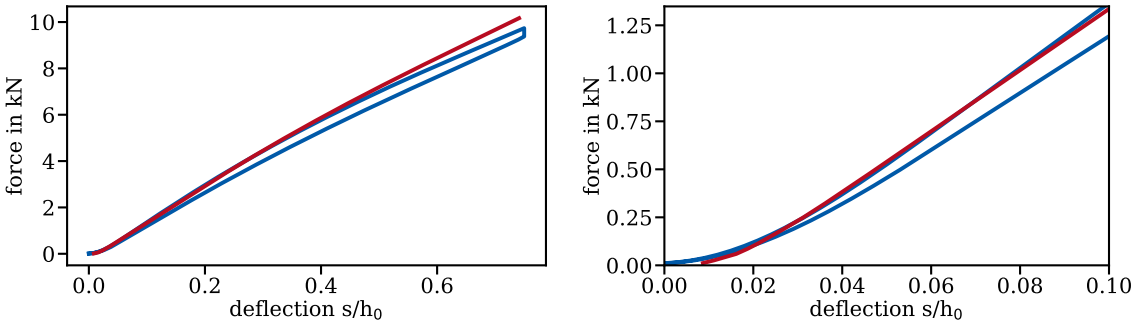

Figure 13. Comparison of characteristics obtained from numerical simulation and from the experiment.

6. Summary

A novel algorithm as implemented in the Spring_stack module was presented in this paper. The algorithm receives geometry data, residual stresses, material parameters, load cases, and further inputs and builds an FE model based on these. It evaluates the FE model after solving with respect to the Walker damage inflicted locally using a Manson–McKnight approach. In a post-processing step, the accumulated surface area as a function of the Walker damage parameter is computed. An example application of the algorithm was presented.

Author Contributions: Conceptualization, M.B.G. and M.K.; methodology, M.B.G., M.K., and M.O.; software, M.B.G.; formal analysis, M.B.G., M.K., and M.O.; resources, M.O.; writing, original draft preparation, M.B.G.; writing, review and editing, M.K. and M.O.; visualization, M.B.G.; supervision, M.O.; project administration, M.K.; funding acquisition, M.O. All authors read and agreed to the published version of the manuscript.

Funding: This article was created as part of the research project AVIF A 309 'Bewertung des Einflusses realer Bauteilgeometrien auf die Beanspruchbarkeit von Tellerfedern anhand numerischer Simulation" (assessment of the influence of real geometries on the load capacity of disc springs by numerical simulation). This project is funded by Stiftung Stahlanwendungsforschung, which is part of Stifterverband für die Deutsche Wissenschaft e.V. (Donors' Association for the Promotion of Science and Humanities in Germany). The Association's mission is the promotion of research into the manufacturing and utilization of steel in Germany. The research proposal was audited by a panel of experts from the Research Association of the Working Group of the Iron- and Metal-processing Industries (AVIF), which is composed of specialists from the steelworking industry and academia. The project is accompanied by a working group from the Association of the German Spring Industry (VDFI).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. DIN EN 16983. *Tellerfedern–Qualitätsanforderungen–Maße*; Beuth: Berlin, Germany, 2017.
2. DIN EN 16984. *Tellerfedern–Berechnung*; Beuth: Berlin, Germany, 2017.
3. Geilen, M.B.; Klein, M.; Oechsner, M. Building Models with 3D-Scanned Geometry using Abaqus Scripting—A Case Study on Disc Springs. In Proceedings of the 3DEXPERIENCE Conference–Design, Modeling & Simulation, Dassault Systemes, Darmstadt, Germany, 19–21 November 2019.
4. Geilen, M.B.; Klein, M.; Oechsner, M. Spring_stack-ein Modul zur numerischen Simulation von Tellerfedern und Tellerfedersäulen. In *Ilmenauer Federntag 2019—Neueste Erkenntnisse zu Funktion, Berechnung, Prüfung und Gestaltung von Federn und Werkstoffen*; ISLE: Ilmenau, Germany, 2019; pp. 77–86.
5. Mischke, C.R.; Shigley, J.E. *Standard Handbook of Machine Design*; McGraw-Hill: New York, NY, USA, 1996.
6. Meissner, M.; Fischer, F.; Wanke, K.; Plitzko, M. *Die Geschichte der Metallfedern und der Federntechnik in Deutschland*; Universitätsverlag Ilmenau: Ilemanu, Germany, 2009.
7. Bhandari, V.B. *Design of Machine Elements*; Tata McGraw-Hill Education: New York, NY, USA, 2010.
8. Curti, G.; Montanini, R. On the influence of friction in the calculation of conical disk springs. *J. Mech. Des.* **1999**, *121*, 622–627. [CrossRef]
9. Bühl, P. Zur Spannungsberechnung von Tellerfedern. *Draht* **1971**, *22*, 760–763.
10. Curti, G.; Orlando, M. Ein neues Berechnungverfahren für Tellerfedern. *Wire* **1979**, *29*, 199–204.
11. Fawazi, N.; Lee, J.Y. An improved load-displacement prediction for a coned disc spring using the energy method. *ARPN J. Eng. Appl. Sci.* **2006**, *11*, 833–836.
12. Hübner, W. Deformationen und Spannungen bei Tellerfedern. *Konstruktion* **1982**, *34*, 387–392.
13. Hübner, W.; Emmerling, F.A. Axialsymmetrische große Deformationen einer elastischen Kegelschale. *ZAMM* **1982**, *62*, 404–406. [CrossRef]
14. Kobelev, V. Exact shell solutions for conical springs. *Mech. Based Des. Struct. Mach.* **2016**, *44*, 317–339. [CrossRef]
15. Lutz, O. Zur Berechnung der Tellerfeder. *Konstruktion* **1960**, *12*, 57–59.
16. Lutz, O.; Wernitz, W. Stülpen einer konischen Ringscheibe (Tellerfeder) unter Berücksichtigung der Mantellinienkrümmung. *Forsch. Ingenieurwesen A* **1967**, *33*, 77–84. [CrossRef]
17. Mastricola, N.P.; Dreyer, J.T.; Singh, R. Analytical and experimental characterization of nonlinear coned disk springs with focus on edge friction contribution to force-deflection hysteresis. *Mech. Syst. Signal Process.* **2017**, *91*, 215–232. [CrossRef]
18. Mastricola, N.P.; Singh, R. Nonlinear load-deflection and stiffness characteristics of coned springs in four primary configurations. *Mech. Mach. Theory* **2017**, *116*, 513–528. [CrossRef]
19. Zheng, E.; Jia, F.; Zhou, X. Energy-based method for nonlinear characteristics analysis of Belleville springs. *Thin-Walled Struct.* **2014**, *79*, 52–61. [CrossRef]
20. Curti, G. Vereinfachtes Verfahren zur Berechnung von Tellerfedern. *Draht* **1980**, *31*, 789–792.
21. Curt, G.; Orlando, M. Geschlitzte Tellerfedern. *Draht* **1981**, *32*, 608–615.
22. Fawazi, N.; Lee, J.Y.; Oh, J.E. A load–displacement prediction for a bended slotted disc using the energy method. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2012**, *226*, 2126–2137. [CrossRef]
23. Fawazi, N.; Yang, I.H.; Kim, J.S.; Lee, J.Y.; Kim, H.S.; Oh, J.E. An inverse algorithm of nonlinear load-displacement for a slotted disc spring geometric design. *Int. J. Precis. Eng. Manuf.* **2013**, *14*, 137–145. [CrossRef]

24. Ferrari, G. A new calculation method for belleville disc springs with contact flats and reduced thickness. *Int. J. Manuf. Mater. Mech. Eng.* **2013**, *3*, 63–73. [CrossRef]
25. La Rosa, G.; Messina, M.; Risitano, A. Stiffness of variable thickness Belleville springs. *J. Mech. Des.* **2001**, *123*, 294–299. [CrossRef]
26. Muhr, K.H.; Niepage, P. Zur Berechnung von Tellerfedern mit rechteckigem Querschnitt und Auflageflächen. *Konstruktion* **1966**, *18*, 24–27.
27. Saini, P.K.; Kumar, P.; Tandon, P. Design and analysis of radially tapered disc springs with parabolically varying thickness. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2007**, *221*, 151–158. [CrossRef]
28. Schremmer, G. Die geschlitzte Tellerfeder. *Konstruktion* **1972**, *24*, 226–229.
29. Boorboor Ajdari, M.; Jalili, S.; Jafari, M.; Zamani, J.; Shariyat, M. The analytical solution of the buckling of composite truncated conical shells under combined external pressure and axial compression. *J. Mech. Sci. Technol.* **2012**, *26*, 2783–2791. [CrossRef]
30. Patangtalo, W.; Aimmanee, S.; Chutima, S. A unified analysis of isotropic and composite Belleville springs. *Thin-Walled Struct.* **2016**, *109*, 285–295. [CrossRef]
31. Kobelev, V. *Durability of Springs*; Springer: Berlin, Germnay, 2018.
32. Ozaki, S.; Tsuda, K.; Tominaga, J. Analyses of static and dynamic behavior of coned disk springs: Effects of friction boundaries. *Thin-Walled Struct.* **2012**, *59*, 132–143. [CrossRef]
33. Schiffner, K.; Borchert, T. Schwingungen flacher Rotationsschalen unter statischer Vorlast. *ZAMM* **1988**, *68* 250-255.
34. Carfagni, M. A CAD program for the automated checkout and design of Belleville springs. *J. Mech. Des.* **2002**, *124*, 393–398. [CrossRef]
35. Paredes, M.; Daidié, A. Optimal catalogue selection and custom design of belleville spring arrangements. *Int. J. Interact. Des. Manuf.* **2010**, *4*, 51–59. [CrossRef]
36. Ha, S.H.; Seong, M.S.; Choi, S.B. Design and vibration control of military vehicle suspension system using magnetorheological damper and disc spring. *Smart Mater. Struct.* **2013**, *22*, 065006. [CrossRef]
37. Wang, W.; Wang, X. Tests, model, and applications for coned-disc-spring vertical isolation bearings. *Bull. Earthq. Eng.* **2020**, *18*, 357–398. [CrossRef]
38. Curti, G.; Appendino, D. Vergleich von Berechnungsverfahren fürTellerfedern. *Draht* **1982**, *33*, 38–40.
39. Wagner, W.; Wetzel, M. Berechnung von Tellerfedern mit Hilfe der Methode der Finiten Elemente. *Konstruktion* **1987**, *39*, 147–150.
40. Bagavathiperumal, P.; Chandrasekaran, K.; Manivasagam, S. Elastic load-displacement predictions for coned disc springs subjected to axial loading using the finite element method. *J. Strain Anal. Eng. Des.* **1991**, *26*, 147–152. [CrossRef]
41. Schiffner, K.; Dietrich, M. Numerische Simulation der Entstehung von Eigenspannungen 1. Art an Tellerfedern. *ZAMM* **1987**, *67*, 235–237.
42. Curti, G.; Raffa, F.A. Material nonlinearity effects in the stress analysis of conical disk springs. *J. Mech. Des.* **1992**, *114*, 238–244. [CrossRef]
43. Yahata, N.; Watanabe, M.; Ishii, N. Analysis of coned disc spring by finite element method. *Jpn. Soc. Mech. Eng.* **1993**, *59*, 260–265.
44. Veleva, D.; Kaiser, B.; Berger, C. Experimentelle Untersuchung und numerische Simulation des Relaxationsverhaltens von Tellerfedern. In *Ilmenauer Federntag 2010—Neueste Erkenntnisse zu Funktion, Berechnung, Prüfung und Gestaltung von Federn und Werkstoffen*; ISLE: Ilmenau, Germany, 2010; pp. 71–78.
45. Veleva, D. Experimentelle Untersuchung und Numerische Simulation des Relaxationsverhaltens von Tellerfedern. Ph.D. Thesis, TU Darmstadt, Darmsatdt, Germnay, 2012.
46. Maqbool, F.; Hajavifard, R.; Walther, F.; Bambach, M. Engineering the residual stress state of the metastable austenitic stainless steel (MASS) disc springs by incremental sheet forming (ISF). *Prod. Eng.* **2019**, *13*, 139–148. [CrossRef]
47. Lee, C.Y.; Chae, Y.S.; Gwon, J.D.; Nam, U.H.; Kim, T.H. Finite element analysis and optimal design of automobile clutch diaphragm spring. *Trans. Korean Soc. Mech. Eng. A* **2000**, *24*, 1616–1623.
48. Karakaya, Ş. Investigation of hybrid and different cross-section composite disc springs using finite element method. *Trans. Can. Soc. Mech. Eng.* **2012**, *36*, 399–412. [CrossRef]

49. Sfarni, S.; Bellenger, E.; Fortin, J.; Malley, M. Numerical and experimental study of automotive riveted clutch discs with contact pressure analysis for the prediction of facing wear. *Finite Elem. Anal. Des.* **2011**, *47*, 129–141. [CrossRef]
50. Yubing, G.; Defeng, Z. Mechanism for the Forced Strengthening on the Diaphragm Spring's Load-Deflection Characteristic. *Int. J. Eng. Technol.* **2017**, *9*, 287–292. [CrossRef]
51. Zhu, D.; Ding, F.; Liu, H.; Zhao, S.; Liu, G. Mechanical property analysis of disc spring. *J. Braz. Soc. Mech. Sci. Eng.* **2018**, *40*, 230. [CrossRef]
52. Chen, Y.S.; Chiu, C.C.; Cheng, Y.D. Dynamic analysis of disc spring effects on the contact pressure of the collet—Spindle interface in a high-speed spindle system. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2009**, *223*, 1191–1201. [CrossRef]
53. Maletta, C.; Filice, L.; Furgiuele, F. NiTi Belleville washers: Design, manufacturing and testing. *J. Intell. Mater. Syst. Struct.* **2013**, *24*, 695–703. [CrossRef]
54. Fang, C.; Zhou, X.; Osofero, A.I.; Shu, Z.; Corradi, M. Superelastic SMA Belleville washers for seismic resisting applications: Experimental study and modeling strategy. *Smart Mater. Struct.* **2016**, *25*, 105013. [CrossRef]
55. Sgambitterra, E.; Maletta, C.; Furgiuele, F. Modeling and simulation of the thermo-mechanical response of NiTi-based Belleville springs. *J. Intell. Mater. Syst. Struct.* **2016**, *27*, 81–91. [CrossRef]
56. Fang, C.; Yam, M.C.H.; Chan, T.M.; Wang, W.; Yang, X.; Lin, X. A study of hybrid self-centring connections equipped with shape memory alloy washers and bolts. *Eng. Struct.* **2018**, *164*, 155–168. [CrossRef]
57. Koplin, C.; Schröder, C.; Becker, W.; Stockmann, J.; Kailer, A. Demonstration der Zuverlässigkeit von keramischen Schraubendruck-und Tellerfedern für korrosive Umgebungen und hohe Temperaturen. In *Ilmenauer Federntag 2019—Neueste Erkenntnisse zu Funktion, Berechnung, Prüfung und Gestaltung von Federn und Werkstoffen*; ISLE: Ilmenau, Germany, 2019; pp. 11–21.
58. Jia, F.; Zhang, F.C. A Study on the Mechanical Properties of Disc Spring Vibration Isolator with Viscous Dampers. *Adv. Mater. Res.* **2014**, *904*, 454–459. [CrossRef]
59. Lonkwic, P.; Różyło, P.; Usydus, I. Numerical Analysis and Experimental Investigation of Disk Spring Configurations with Regard to Load Capacity of Safety Progressive Gears. *Appl. Comput. Sci.* **2016**, *12*, 5–16.
60. Zhou, J.; Zhang, C.; Wang, Z.; Mao, K. Study on Dynamic Characteristics of the Disc Spring System in Vibration Screen. *Shock Vib.* **2020**, *2020*, 3518037. [CrossRef]
61. Kletzin, U. Finite Elemente basiertes Entwurfssystem für Federn und Federanordnungen. Ph.D. Thesis, TU Ilmenau: Ilmenau, Germany, 2000.
62. Wehmann, C.; Neidnicht, M.; Nützel, F.; Roith, B.; Rieg, F. Finite Elemente Analysen zur Berechnung von Maschinenelementen mit nichtlinearem Verhalten. In *8. Gemeinsames Kolloquium Konstruktionstechnik—Herausforderungen für die Produkt-und Prozessinnovation*; Docuprint: Barleben, Germany, 2010; pp. 47–52.
63. Walker, K. The effect of stress ratio during crack propagation and fatigue for 2024-T3 and 7075-T6 aluminum. In *Effects of Environment and Complex Load History on Fatigue Life*; ASTM International: West Conshohocken, PA, USA, 1970.
64. Goodman, J. *Mechanics Applied to Engineering*; Longmans, Green: Harlow, UK, 1919.
65. Morrow, J. Fatigue properties of metals. In *Fatigue Design Handbook*; Society of Automotive Engineers: Warrendale, PA, USA, 1968; pp. 21–30.
66. Smith, K.N.; Watson, P.; Topper, T.H. A Stress-Strain Function for the Fatigue of Metals. *J. Mater.* **1970**, *5*, 767–778.
67. Dowling, N.E.; Calhoun, C.A.; Arcari, A. Mean stress effects in stress–life fatigue and the Walker equation. *Fatigue Fract. Eng. Mater. Struct.* **2009**, *32*, 163–179. [CrossRef]
68. Dowling, N.E. *Mechanical Behavior of Materials: Engineering Methods for Deformation, Fracture, and Fatigue*; Pearson: London, UK, 2012.
69. Gyekenyesi, J.Z.; Murthy, P.L.N.; Mital, S.K. *NASALIFE-Component Fatigue and Creep Life Prediction Program*; NASA Glenn Research Center: Cleveland, OH, USA, 2014.
70. Dassault Systemes. *Abaqus 2018 Scripting User's Guide*; Dassault Systemes: Vélizy-Villacoublay, France, 2018.
71. Dassault Systemes. *Abaqus 2018 Scripting User's Reference Guide*; Dassault Systemes: Vélizy-Villacoublay, France, 2018.
72. Barber, C.B.; Dobkin, D.P.; Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.* **1996**, *22*, 469–483. [CrossRef]

73. Virtanen, P.; Gommers, R.; Oliphant, T.E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv* **2019**, arXiv:1907.10121.
74. Byrd, R.H.; Lu, P.; Nocedal, J.; Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.* **1995**, *16*, 1190–1208. [CrossRef]
75. Zhu, C.; Byrd, R.H.; Lu, P.; Nocedal, J. L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.* **1997**, *23*, 550–560. [CrossRef]
76. Cignoni, P.; Callieri, M.; Corsini, M.; Dellepiane, M.; Ganovelli, F.; Ranzuglia, G. Meshlab: An open-source mesh processing tool. In Proceedings of the Eurographics Italian Chapter Conference, Salerno, Italy, 2–4 July 2008; pp. 129–136.
77. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
78. Dassault Systemes. *Abaqus 2018 Analysis User's Guide*; Volume IV: Elements; Dassault Systemes: Vélizy-Villacoublay, France, 2018.
79. Spies, A.; Beyer, J.; Oechsner, M. Ermittlung und Bewertung der Schwingfestigkeitseigenschaften von Tellerfedern aus verschiedenen Werkstoffen. In *Ilmenauer Federntag 2015—Neueste Erkenntnisse zu Funktion, Berechnung, Prüfung und Gestaltung von Federn und Werkstoffen*; ISLE: Ilmenau, Germany, 2015; pp. 121–130.
80. Hertzer, K.H. Über die Dauerfestigkeit und das Setzen von Tellerfedern; Ph.D. Thesis, Technische Hoschschule Carolo-Wilhelmina: Braunschweig, Germany, 1959.
81. Muhr, K.H.; Niepage, P.; Willwacher, H. Warmvorsetzen vermindert die Relaxation von Tellerfedern. *Konstruktion* **1975**, *27*, 468–471.
82. Niepage, P.; Grahn, B.; Virnich, K.H. Der Einfluss des Eigenspannungszustandes auf die Schwingfestigkeit von Tellerfedern. *Draht* **1993**, *44*, 224–227.
83. Buchfink, B.; Naue, K.C.; Wunderle, G. *Tellerfedern: Funktion, Fertigung, Werkstoffe*; Die Bibliothek der Technik, Verlag Moderne Industrie: Landsberg am Lech, Germany, 2015.
84. Christian Bauer GmbH & Co. KG. *Tellerfedern–Theorie und Praxis*.
85. Mubea Disc Scprings. *Mubea Disc Springs Manual*.