Bedienungsanleitung EEHBV

EEHB ist eine Webapplikation, die der Nutzer im Browser ausführen kann. Jede aktuelle Browser-Version mit eingeschalteter JavaScript-Unterstützung sollte dafür geeignet sein. Für die Installation befolgen Sie bitte die im beiliegenden Dokument Installation.pdf beschriebenen Schritte und rufen Sie danach die URL im Browser auf.

Inhalt

[Anwendungsbereich 3](#_Toc150124513)

[Benutzertypen 4](#_Toc150124514)

[Ansichten 4](#_Toc150124515)

[Komponenten 4](#_Toc150124516)

[Materialeigenschaften 5](#_Toc150124517)

[Problemklassen 6](#_Toc150124518)

[Historie 8](#_Toc150124519)

[Glossar 8](#_Toc150124520)

[Nutzerverwaltung 8](#_Toc150124521)

[Anwendungsfälle 9](#_Toc150124522)

[Optimierungsanfrage 9](#_Toc150124523)

[Betrachten von bereits ausgeführten Optimierungsanfragen 13](#_Toc150124524)

[Anlegen/Bearbeiten/Importieren von Komponentendaten 13](#_Toc150124525)

[Anlegen von Komponententypen 14](#_Toc150124526)

[Anlegen/Bearbeiten von Materialeigenschaften 14](#_Toc150124527)

[Anlegen/Bearbeiten von Glossareinträgen 14](#_Toc150124528)

[Anlegen/Bearbeiten von Nutzern 14](#_Toc150124529)

[Definition neuer Problemklassen 14](#_Toc150124530)

[Vorgehensrahmenmodell 15](#_Toc150124531)

[Identifikation der Komponententypen/Aggregate 15](#_Toc150124532)

[Identifikation und Systematisierung der Varianten 15](#_Toc150124533)

[Modellierung der Prozess- und Verlustleistungen der einzelnen Prozessschritte als Funktion 18](#_Toc150124534)

[Auflistung der Variablen 18](#_Toc150124535)

[Identifikation impliziter Nebenbedingungen 19](#_Toc150124536)

[Entwurf Auswahlalgorithmus 25](#_Toc150124537)

[Datenbankmodell für Komponenten anlegen und mit Daten befüllen 26](#_Toc150124538)

[Materialeigenschaften in Datenbank anlegen 26](#_Toc150124539)

[Anlegen des Prozessmodells in der Anwendung als Problemklasse 26](#_Toc150124540)

[Funktionen für Beispiel-Prozess 30](#_Toc150124541)

[Python-Code für Prozessleistung rotierende Frästools (process\_power\_scraper.py) 30](#_Toc150124542)

[Python-Code für am Motor abfallende Leistung (loss\_motor.py) 36](#_Toc150124543)

[Python-Code für die am Spannungswandler abfallende Wärmeleistung (loss\_converter.py) 37](#_Toc150124544)

[Python-Code für Getriebe-Leistungen und –momente (gears.py) 38](#_Toc150124545)

[Python-Code für Reibungsverluste am Vortrieb (feed.py) 38](#_Toc150124546)

[Python-Code für Zwischengrößen für Kappaggregat (trim\_values.py) 39](#_Toc150124547)

[Python-Code für Prozessleistung an Abziehklingen (process\_power\_scraper.py) 39](#_Toc150124548)

[Python-Code für Wäremeleistung für Gleichgewichtszustand Kleberbehälter (glue.py) 40](#_Toc150124549)

[Python-Code für Kantenbandbreite am Bündigfräsaggregat (overlap.py) 40](#_Toc150124550)

[Python-Code Hilfsfunktion zur Parameterauswahl (select\_from\_list.py) 40](#_Toc150124551)

# Anwendungsbereich

Die Software soll für die Konstruktion von Sondermaschinen eine auf den konkreten Anwendungsbedarf des Nutzers zugeschnittene Konstellation von Maschinenkomponenten auswählen, die auf einen möglichst energie-effizienten Betrieb optimiert ist. Der Auswahlprozess stellt sicher, dass vom Nutzer vorgegebene Rahmenbedingungen erfüllt werden. Die möglichen Komponenten werden aus einer integrierten Datenbank ausgewählt, die vom Nutzer der Software beliebig um neue Komponententypen und Einträge erweiterbar ist.

Um diesen Auswahlprozess für einen bestimmten Typ von Sondermaschinen durchführen zu können, muss von einem Fachanwender für den Maschinentyp zunächst ein Energie-Leistungs-Modell für die beteiligten Prozesse und Komponenten angelegt und in der Software hinterlegt werden. Ein Vorgehensmodell für die Erzeugung der notwendigen Modelle wird im Kapitel Vorgehensrahmenmodell beschrieben, die konkrete menügestützte Eingabe des Modells in die Software in Abschnitt Definition neuer Problemklassen. Da Sondermaschinen oft in verschiedenen Bauformen vorkommen können, unterstützt das System die Anlage von Varianten zu einem Maschinentyp. Über diese Varianten lässt sich zum Beispiel abbilden, ob bestimmte Aggregate optional oder in unterschiedlichen technischen Umsetzungen ausführbar sind.

Für hinterlegte Maschinentypen können Auslegungsanfragen menügeführt durchgeführt werden (Abschnitt Optimierungsanfrage), wobei jeweils spezifische Nutzprofile und Randbedingungen angegeben werden müssen.

Da für bestimmte Prozesse die Prozessleistungen von Eigenschaften zu bearbeitender Werkstoffe abhängt, gibt es in der Software auch die Möglichkeit, diese Eigenschaften für bestimmte Materialien in der integrierten Datenbank zu hinterlegen und dann bei einer Auslegungsanfrage auf die eingetragenen Werte zuzugreifen.

Darüber hinaus können die Anwender bereits gestellte Anfragen wieder einsehen und in einem Glossar selber eine erweiterbare Dokumentation zu den eigenen Anwendungszwecken pflegen.

# Benutzertypen

Die Anwendung unterscheidet zwischen vier Typen von Anwendern:

* Gäste
* Datenberechtigte (Benutzerrolle data)
* Problemklassenberechtigte (Benutzerrolle opt)
* Administratoren (Benutzerrolle admin)

Die weiter unten Stehenden in dieser Liste haben dabei jeweils die Rechte der Vorgänger sowie zusätzliche weitere.

**Gäste** können sich ohne Anmeldung alle Haupt-Bereiche der Anwendung (bis auf die Nutzerverwaltung) anschauen und können Optimierungsanfragen stellen.

**Datenberechtigte** können Komponenten, Materialeinträge sowie Glossareinträge anlegen, ändern und löschen.

**Problemklassenberechtigte** können neue Komponenten- und Materialeigenschaftstypen und neue Problemklassen anlegen.

**Administratoren** können Benutzer verwalten.

Benutzer können sich über das Nutzer-Symbol rechts in der horizontalen Menüleiste am oberen Fensterrand anmelden und auch wieder abmelden.

# Ansichten

Bis auf die Nutzerverwaltung sind alle Hauptansichten für alle Anwendertypen inklusive Gästen sichtbar. Sollte die vertikale Menüleiste am linken Fensterrand nicht sichtbar sein, können sie diese über durch einen Klick auf das Symbol mit den drei Balken links in der horizontalen Menüleiste am oberen Fensterrand einblenden. Gegebenenfalls empfiehlt es sich, diese Leiste nach Anwahl der gewünschten Hauptansicht über das gleiche Symbol wieder auszublenden.



Abbildung 1: Startbildschirm mit vertikaler Menüleiste links, Button für die Ein-/Ausblendung der Menüleiste (oben mittig, gelb markiert), und Anmeldebutton (oben rechts).

## Komponenten

In der Hauptansicht für Komponenten sind alle hinterlegten Komponententypen sichtbar und anwählbar.

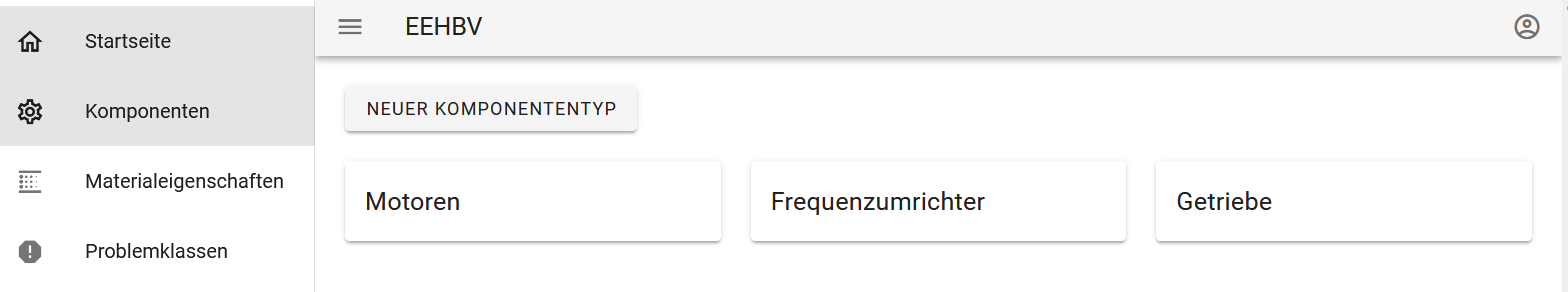


Abbildung 2: Hauptansicht Komponenten, Übersicht von Komponententypen

Die Funktion zum Anlegen neuer Komponententypen ist nur ab der Rolle opt verfügbar. Nutzer ab der Rolle data können in der Detailansicht eines Komponententyps neue Einträge einfügen, sowie über CSV-Dateien importieren und bestehende verändern und löschen. Sowohl für manuelle Einträge als auch beim Dateiimport müssen für alle Spalten Werte angegeben werden.

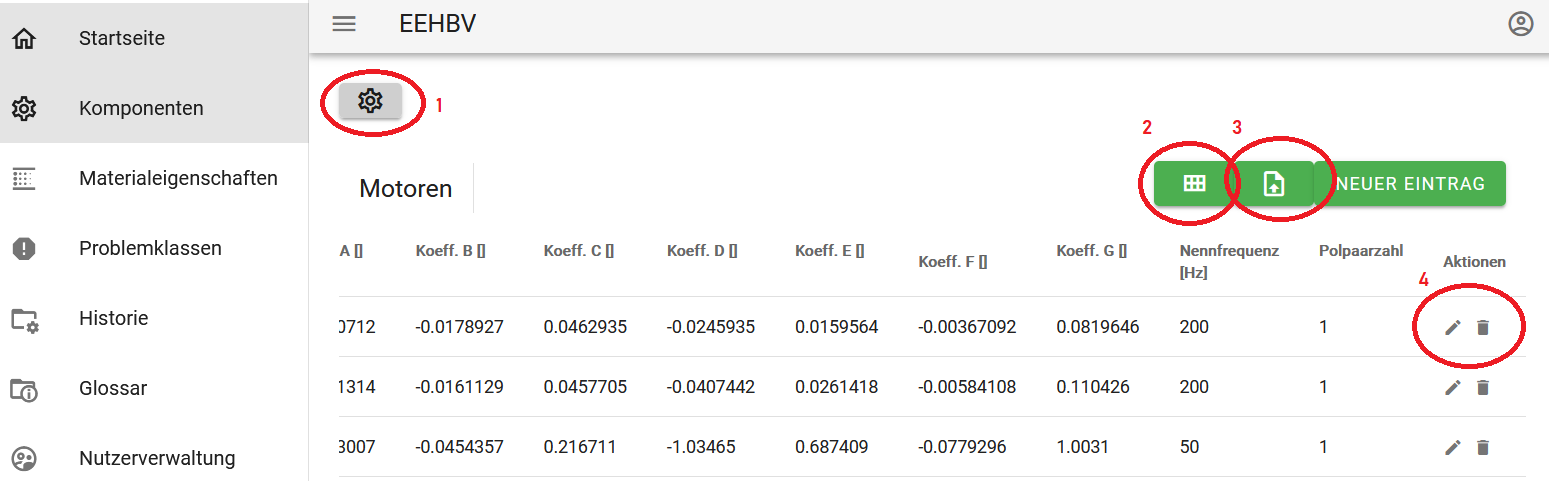


Abbildung 3: Tabellendarstellung eingetragener Motor-Daten - die rot markierten Buttons navigieren zu folgenden Funktionen: 1: zurück zur Komponententypenüebrsicht; 2: Anzeige der Datenbankspalten (notwendig für Dateiimport), 3: Datei-Upload; 4: Bearbeiten und Löschen von tabelleneinträgen

Beim Dateiimport über eine CSV-Datei muss die zu importierende Datei mit kommaseparierten Spalten formatiert sein. Als Dezimaltrennzeichen sind Kommata zu nutzen. Die Datei muss in der ersten Zeile die Spaltennamen der entsprechenden Datenbanktabelle enthalten. Diese Spaltennamen können in der Tabellenansicht der Komponentendaten über den Button mit dem Tabellensymbol im Kopf der Tabelle angezeigt werden. Die zu importierende Datei muss alle Spalten (in beliebiger Reihenfolge) enthalten.

## Materialeigenschaften

In der Ansicht werden verschiedene Typen von Materialeigenschaften dargestellt und verwaltet. Der Typ der anzuzeigenden Materialeigenschaften kann über ein Dropdown-Menü ausgewählt werden. Ab der Nutzerrolle data können über den +-Button direkt daneben neue Typen von Materialeigenschaften angelegt werden, über den Button „NEUER EINTRAG“ oben rechts für den ausgewählten Typ neue Einträge für spezifische Materialien.

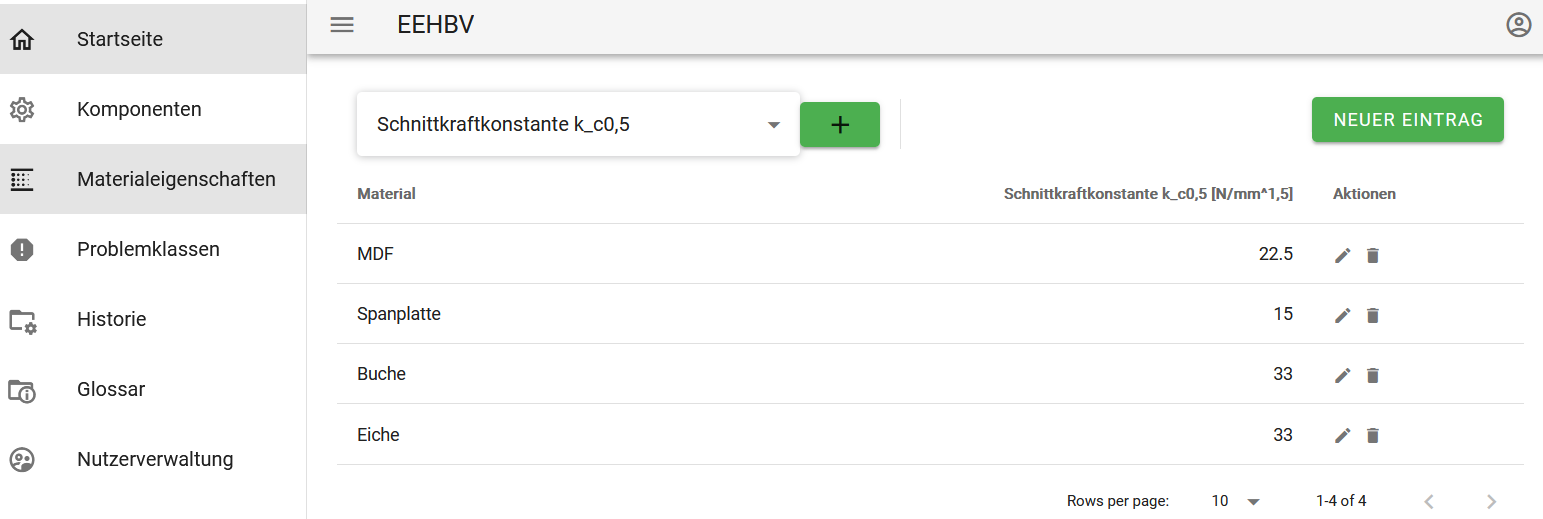


Abbildung 4: Ansicht zur Verwaltung von Materialeigenschaften

## Problemklassen

Als Problemklassen werden in der Software Definitionen von Maschinentypen verstanden, für die die Software Optimierungen vornehmen kann. Diese Definitionen sind vom Nutzer der Software zunächst anzulegen, bevor für eine konkrete Auslegung einer Maschine von diesem Typ mit Anforderungsprofil optimiert werden kann. Die Anlage neuer Prozesstypen wird eingehend im Abschnitt Vorgehensrahmenmodell beschrieben.

Unter dem Menüpunkt Problemklassen findet sich in der Hauptansicht eine Auflistung bereits angelegter Prozesstypen sowie für Nutzer mit Problemklassen-Berechtigung die Buttons „NEUER PROZESSTYP“ und „JSON-UPLOAD PROZESSTYP“.

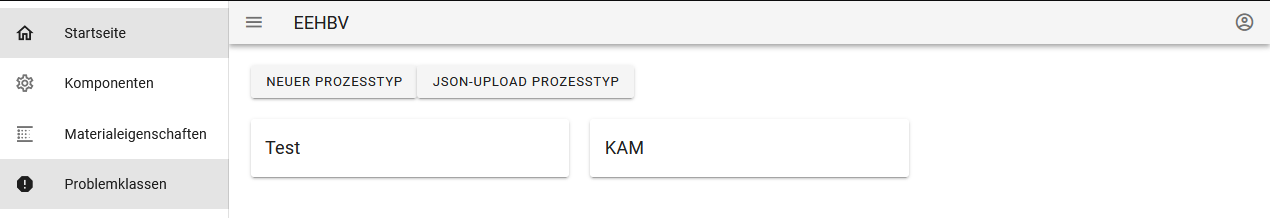


Abbildung 5: Hauptansicht der Problemklassen mit zwei angelegten Prozesstypen

Über ersteren erfolgt eine menügeführte Anlage eines neuen Prozesstyps, die zuvor eingehend vorbereitet sein muss. Wenn man ein tiefgehendes Verständnis der dabei im System abgelegten Datenstruktur hat, kann die Anlage auch über den Upload eines im JSON-Format vorbereiteten Modells erfolgen.

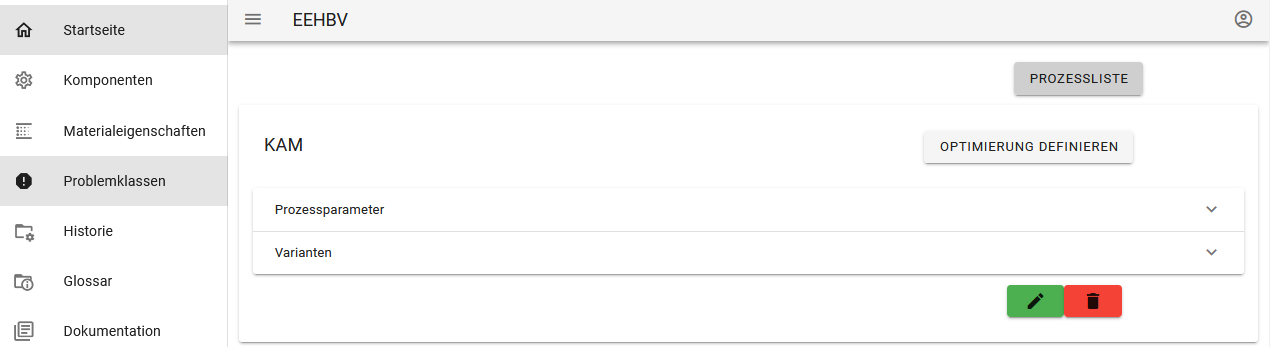


Abbildung 6: Details-Ansicht einer angelegten Problemklasse mit zugeklappten Übersichten für die Prozessparameter und Varianten.

Über die Auswahl einer Problemklasse kommt man zur Übersichtsansicht für diese, in der die Prozessparameter und Varianten aufgelistet sind. Nutzer mit Problemklassen-Berechtigung können die Definition der Problemklasse bearbeiten oder löschen (grüner und roter Button in Abbildung 6). Jeder Nutzer kann über den Button „OPTIMIERUNG DEFINIEREN“ eine Optimierungsanfrage für eine neu zu projektierende Maschine stellen.

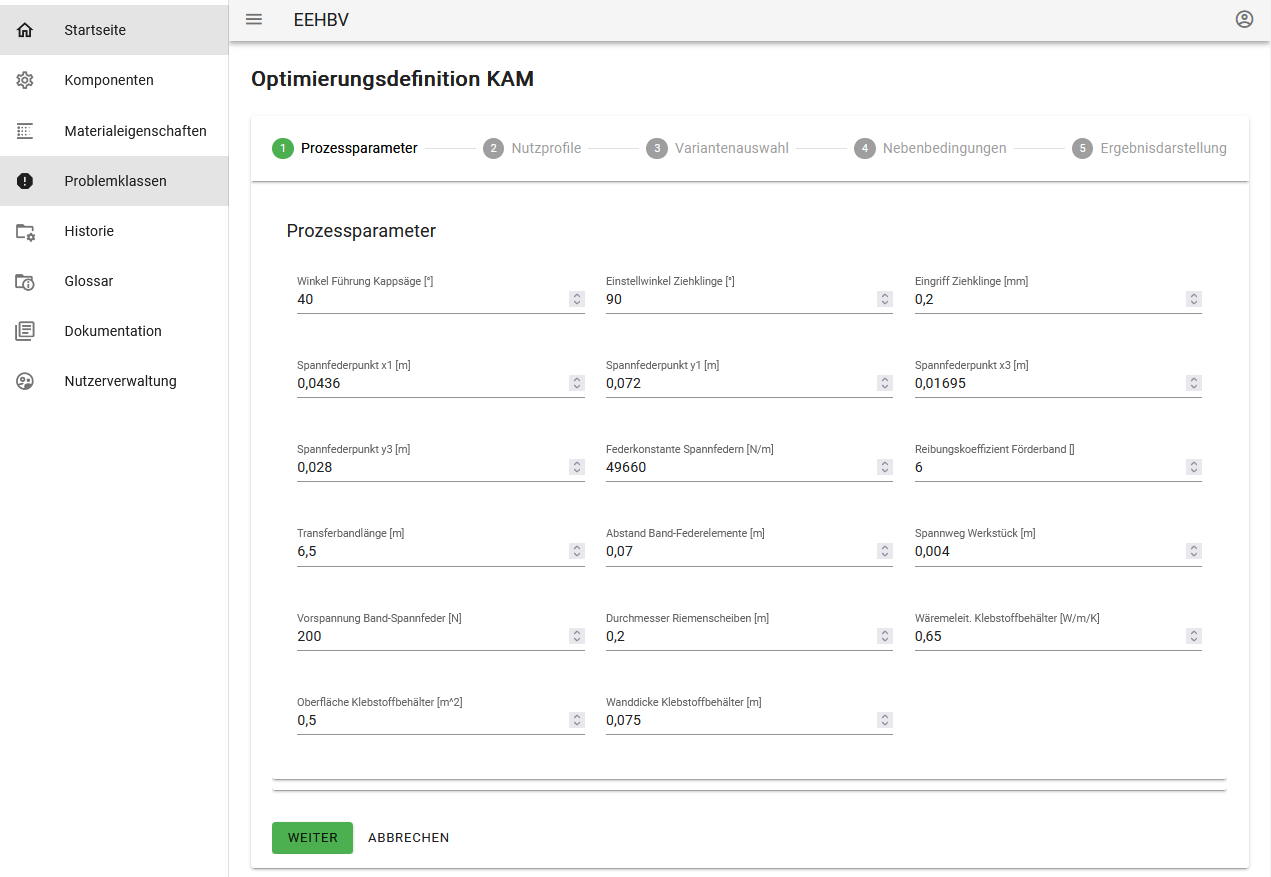


Abbildung 7: Menügeführte Definition der Anforderungen bei einer Optimierungsanfrage

Dabei wird vom Nutzer das Anforderungsprofil der neu zu entwerfenden Maschine in mehreren Formularen abgefragt. Dies wird im Detail im Abschnitt Optimierungsanfrage beschrieben.

Nach Absenden der Optimierungsanfrage wechselt das Programm nach erfolgter Berechnung in die Ergebnisansicht. Auch diese wird im Abschnitt Optimierungsanfrage erläutert.

In der Ergebnisansicht kann man über den Button MASCHINE UNTERSUCHEN in eine Ansicht wechseln, in der man die Parameterbereiche der Komponentenzusammenstellung austesten und eine Parameteroptimierung für einen konkreten Betriebsfall vornehmen kann.

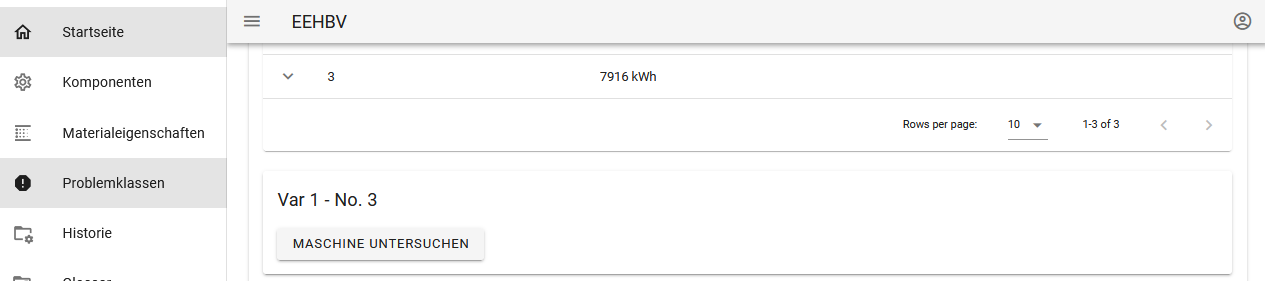


Abbildung 8: Aus der Ergebnisansicht kann man über den Button MASCHINE UNTERUSCHEN in eine Parameter-Optimierung für die Maschine wechseln.

## Historie

In der Hauptansicht Historie können bereits gestellte Optimierungsanfragen für die erneute Ergebnisanzeige ausgesucht werden. Dafür muss eine Problemklasse ausgewählt und über den Button ANZEIGEN bestätigt werden.

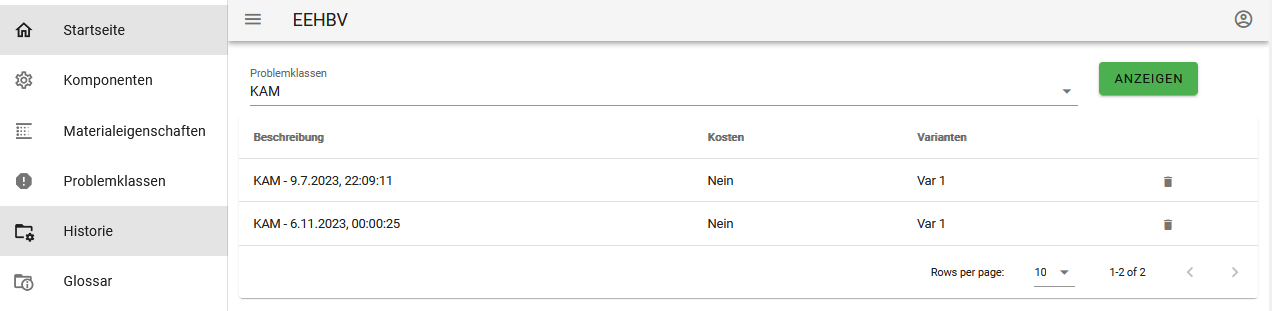


Abbildung 9: Die Anzeige bereits erfolgter Optimierungsanfragen kann über den Menüpunkt Historie erfolgen.

## Glossar

Unter dem Menüpunkt Glossar können Begriffserklärungen hinterlegt werden, die für das Verständnis einer oder mehrerer angelegter Problemklassen hilfreich sind. Die Anzeige der Einträge kann nach Problemklassen gefiltert werden.

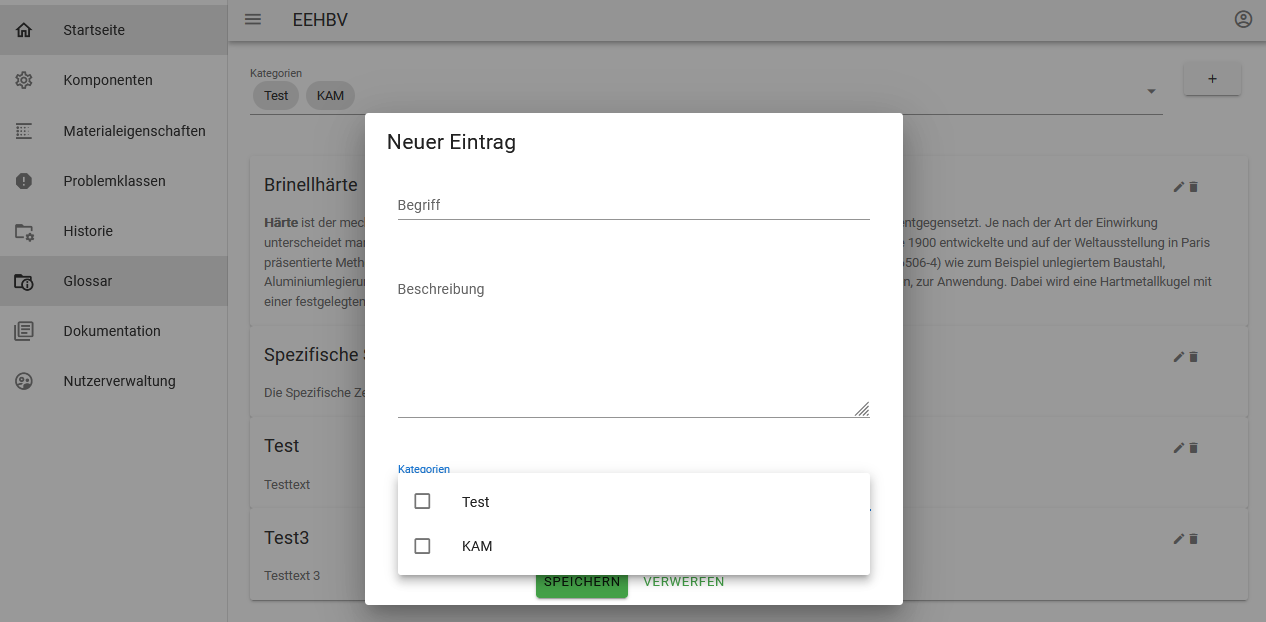


Abbildung 10: Die Einträge des Glossars können mehreren Problemklasse zugeordnet und die Anzeige nach diesen gefiltert werden.

## Nutzerverwaltung

Normale angemeldete Benutzer können in der Nutzerverwaltung ihr Passwort ändern. Administratoren können zusätzlich neue Nutzer anlegen, löschen oder ihre Rolle und Passwort ändern.

# Anwendungsfälle

## Optimierungsanfrage

Der Hauptanwendungszweck der Software besteht darin, konkrete Optimierungsanfragen für einen Maschinentyp an das System zu stellen.

Dazu wählen Sie in der Hauptansicht Problemklassen den entsprechenden Maschinentyp aus und betätigen den Button „OPTIMIERUNG DEFINIEREN“. Die Ansicht wechselt in schrittweise auszufüllende Formulare, in denen zunächst die gewünschten Parameter für den Anwendungsfall, danach – falls im hinterlegten Maschinenmodell vorhanden – zu berücksichtigende Varianten, eventuelle Nebenanforderungen an den Prozess, sowie schließlich einige Angaben zur gewünschten Ergebnisberechnung und -darstellung eingetragen werden müssen.

Im ersten Schritt werden generelle Parameter abgefragt, die im Betrieb der Maschine nicht mehr verändert werden können.

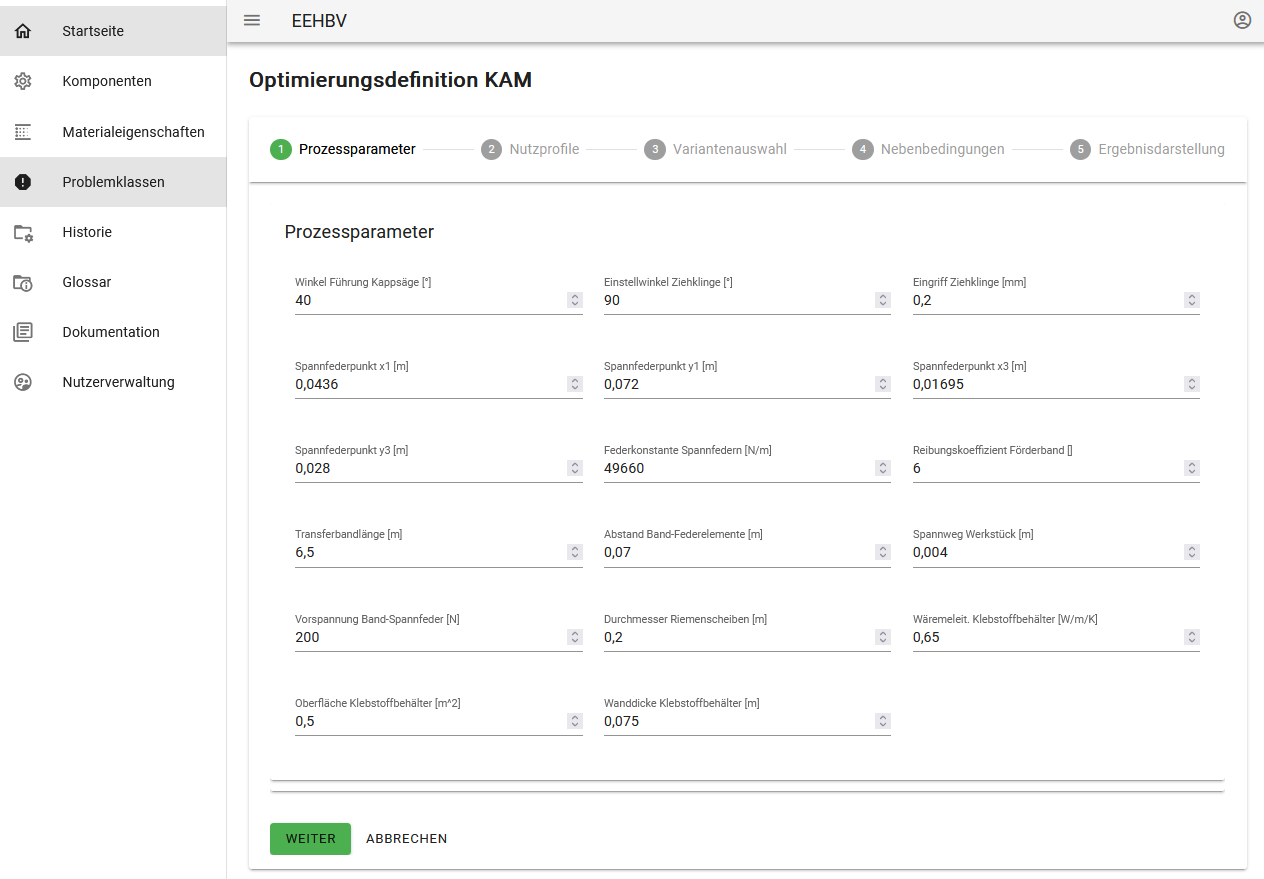


Abbildung 11: Formular zu Abfrage der generellen Parameter für Kantenanleimmaschinen. Diese Parameter sind im Betrieb der Maschine nicht veränderbar.

Im zweiten Schritt können mehrere Nutzungsprofile angegeben werden, mit denen die Maschine betrieben werden soll. Konkret ist dies zur Leistungsbetrachtung beim Produzieren unterschiedlicher Bauteile notwendig, da sich die Prozessleistungen und Wärmeverluste in Abhängigkeit von der konkreten Auslastung unterscheiden. Für die Optimierung ist wesentlich, dass eine Abschätzung darüber getroffen werden kann, in welchem Umfang das konkrete Profil genutzt wird (anzugeben in Stunden pro Jahr). Wichtig ist auch, dass Nutzungsprofile berücksichtigt werden, die vielleicht recht selten zum Einsatz kommen sollen, aber z. B. durch besondere Werkstückeigenschaften hohe Lastanforderungen an die Maschine stellen werden. Dies ist notwendig, damit auch für diese Anwendungsfälle ausreichend leistungsstarke Komponenten bei der Auswahl berücksichtigt werden. Neben der Nutzungsdauer sind in den Profilen Parameter einzugeben, die von den Werkstücken oder Werkstoffen abhängig sind oder bestimmte Betriebsmodi der Maschine oder Eigenschaften von wechselbaren Werkzeugen beschreiben.

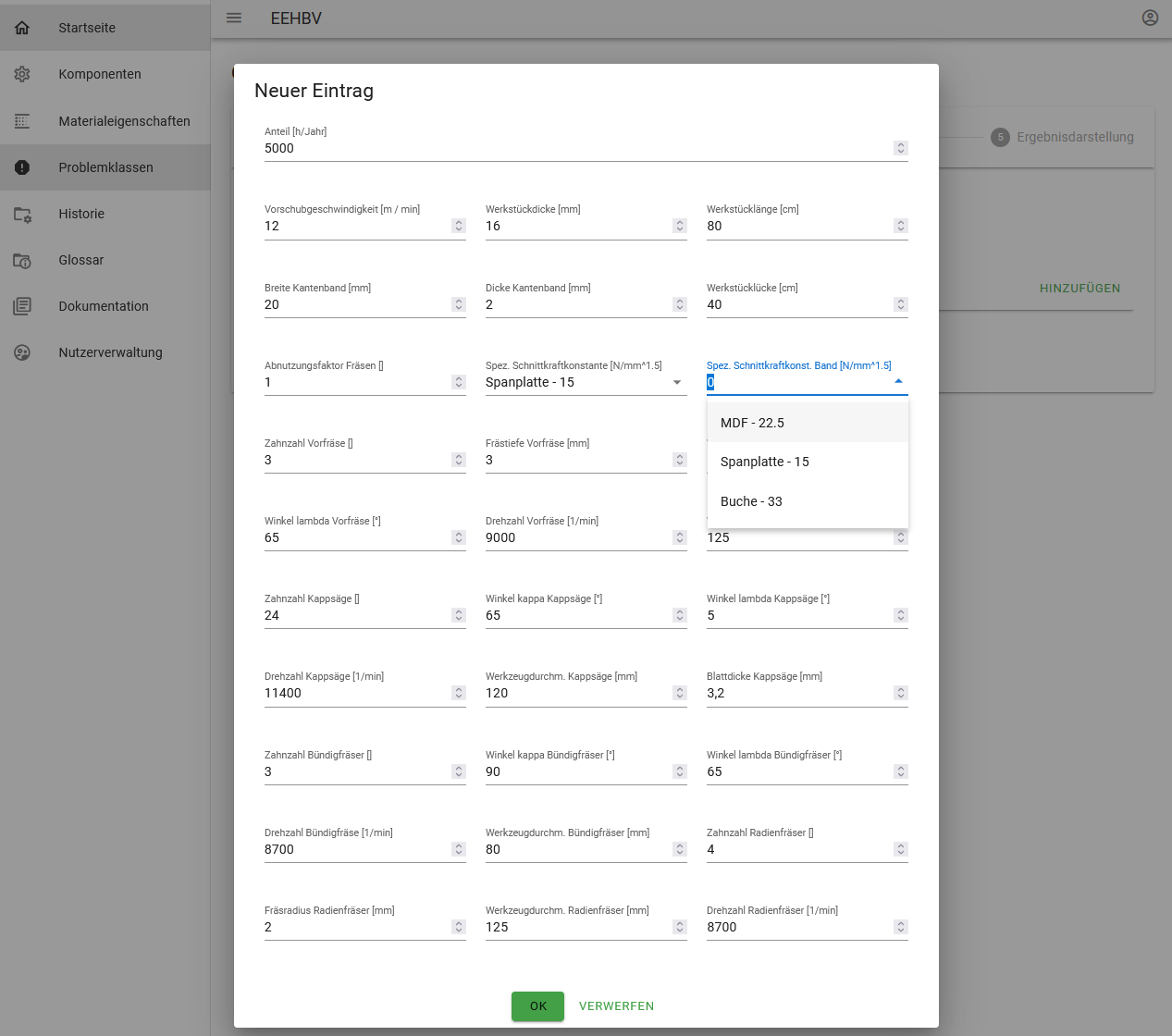


Abbildung 12: Definition eines Nutzprofils. Für verschiedene Anforderungen (z. B. zu produzierende Bauteile) kann jeweils ein Nutzprofil mit der geschätzten Nutzungsdauer in diesem Profil angelegt werden.

Sind für den Maschinentyp mehrere Varianten hinterlegt, ist im nächsten Schritt auszuwählen, welche Varianten für die Optimierung berücksichtigt werden sollen. Die (bis zu vier) ausgewählten Varianten werden bei der Optimierung unabhängig voneinander betrachtet und im Ergebnis zum Vergleich nebeneinander dargestellt. Grundsätzlich steht eine Auswahl über eine Pickliste zur Verfügung. Alternativ kann die Auswahl über eine Abfolge von Fragen getroffen werden, die die benötigten Prozessschritte oder technische Umsetzung berücksichtigen. Die Variantenauswahl wird sich daran orientieren, ob für alle geplanten Nutzungsfälle der Maschine bestimmte Teilprozesse notwendig sind oder entfallen können. Berücksichtigen die Varianten verschiedene technische Umsetzungen von Teilprozessen, kann es sein, dass bestimmte Varianten in der Ausführung durch die zu verwendenden Komponenten grundsätzlich teurer sein werden. Hier wird sich die Wahl der zu betrachtenden Varianten zusätzlich am zur Verfügung stehenden Budget orientieren.

Getrennt für die ausgewählten Varianten können im nächsten Schritt jeweils weitere Einschränkungen an die auszuwählenden Komponenten eingetragen werden, die sich nicht bereits durch die Anforderungsprofile ergeben. Eine Einschränkung kann zum Beispiel sein, dass die Summe der Preise aller ausgewählten Komponenten einen bestimmten Wert nicht übersteigt. Die Einschränkungen müssen in Form von Ungleichungen angegeben werden. Für die Eingabe steht hier ein Editor bereit, der die zur Verfügung stehenden Prozessparameter und Komponenteneigenschaften sowie erlaubte Operatoren und Funktionen bereitstellt, die in diese Ungleichungen eingehen können. Zu beachten ist, dass die Gleichungen nur in einseitiger Form angegeben werden können, die andere Seite also immer zu 0 hin aufgelöst sein muss. Eine Konsistenzüberprüfung der Einheiten in den angegebenen Gleichungen findet bei der Eingabe nicht statt. Darauf muss der Anwender selber achten.

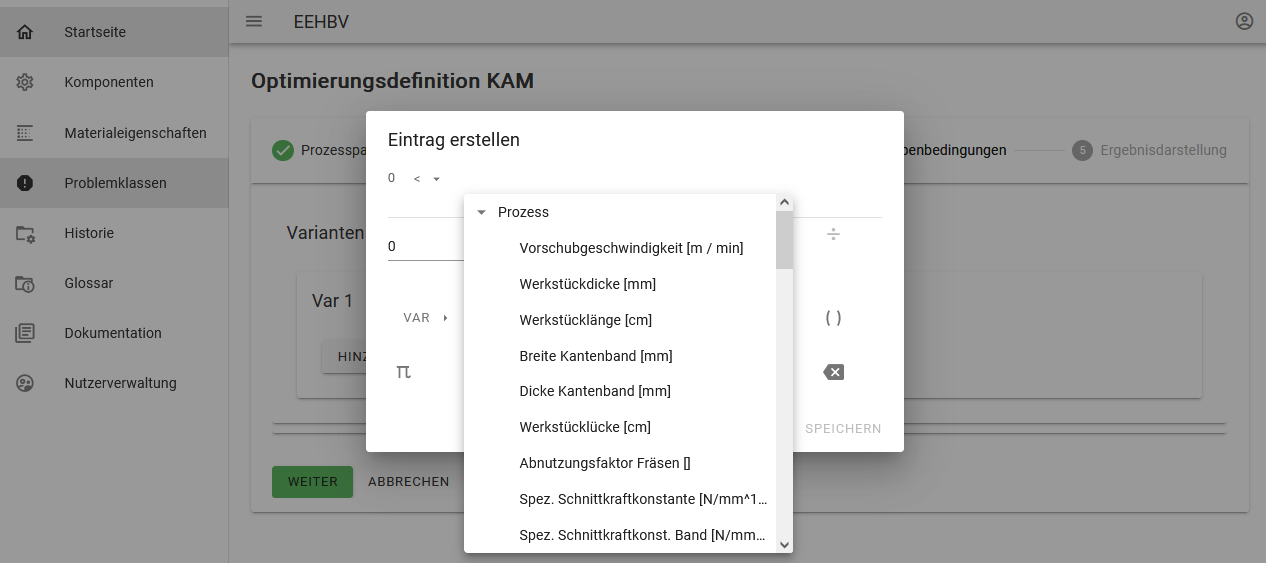


Abbildung 13: Die Definition von Nebenbedingungen erfolgt bei der Optimierungsanfrage über einen Formeleditor, bei dem die Prozessparameter über einen Drop-Down-Button in die Ungleichung eingefügt werden können.

Im letzten Schritt ist anzugeben, wie viele Ergebnisse die Optimierung zurückliefern soll. Es werden jeweils die Konstellationen von Komponenten zurückgeliefert, die für die angegebenen Nutzungsprofile für geringsten Energieverbrauch sorgen. An dieser Stelle kann auch angegeben werden, ob eine Kostenbetrachtung durchgeführt werden soll und die Optimierung unter Berücksichtigung von Energiekosten und Anschaffungskosten durchgeführt. Dafür müssen aber die Preise der in der Datenbank hinterlegten Komponenten mit aktuellen Daten gepflegt sein und zusätzlich der zu erwartende Energiepreis pro kWh und ungefähre Zusatzkosten beim Bau der Maschine (jenseits der verwendeten Komponenten) angegeben werden.

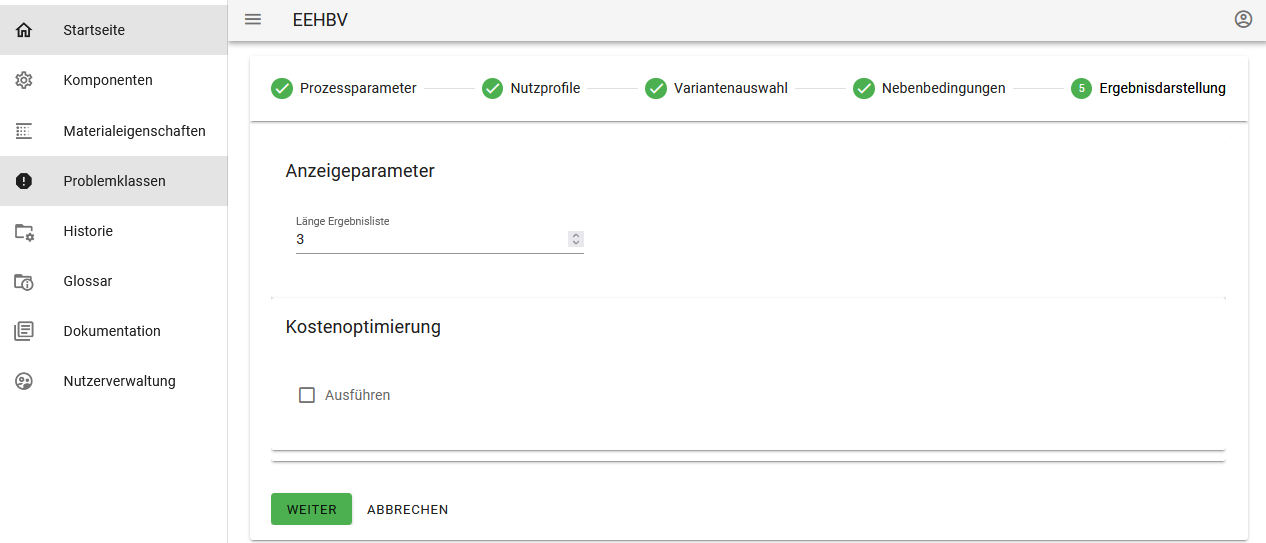


Abbildung 14: Bei der Optimierungsanfrage kann angegeben werden, wie viele optimierte Ergebnisse zurückgeliefert werden sollen.

Nach Absenden der der Optimierungsanfrage dauert es einige Sekunden bis Minuten, bis das Ergebnis im Browser angezeigt wird. Die Dauer hängt wesentlich von der Anzahl der in der Datenbank hinterlegten Komponenten ab, da diese auf das energieoptimale Ergebnis hin verglichen werden müssen. Bis die Anzeige vorliegt, wird auf der Seite unter der Zusammenfassung der Anfrage ein Ladesymbol angezeigt.

In der Ergebnisübersicht wird untereinander die zuvor ausgewählte Anzahl an Vorschlägen der Komponentenkonstellationen mit ihren Jahresenergieverbräuchen aufgelistet. Dabei ist der energieoptimale Vorschlag ganz oben aufgeführt. Wurden mehrere Varianten zum Vergleich ausgewählt, stehen jeweils die Vorschläge der Varianten in ihrer Rangfolge nach Energieeffizienz nebeneinander. Details zu den Komponenten und den Einzelverbräuchen können durch Aufklappen der jeweiligen Zeile über das Pfeilsymbol angezeigt werden.

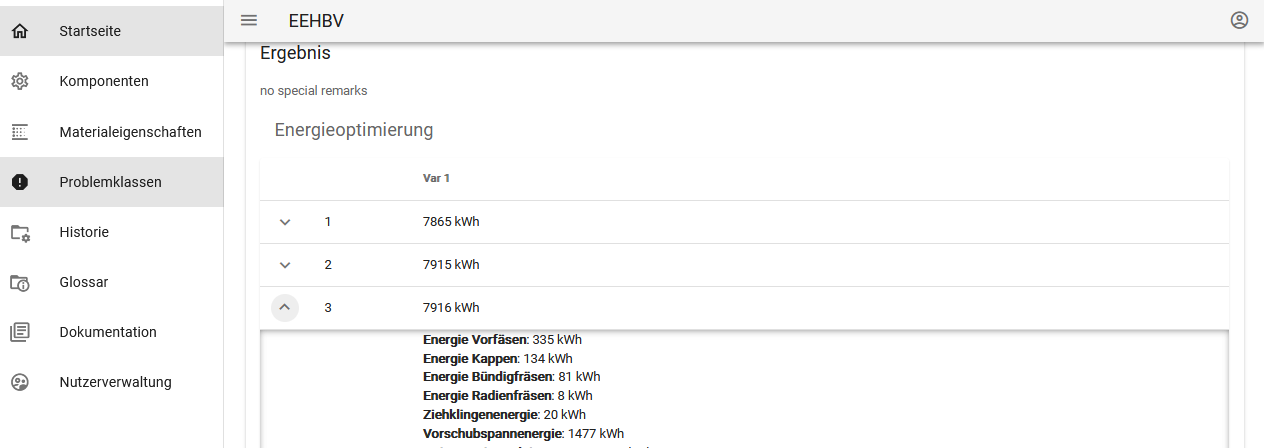


Abbildung 15: Ausschnitt aus der Ergebnisanzeige für eine Optimierungsanfrage. Neben den Energieverbräuchen der Teilaggregate wird die Liste aller Komponenten ausgegeben, die die hinsichtlich Energieeffizienz optimalen Ergebnisse für die Zusammenstellung einer Maschine liefern.

Wurden mehrere Nutzungsprofile für die Optimierung angegeben, befindet sich unter der Übersicht eine tabellarische Darstellung der Teilverbräuche für alle Profile.

Darunter befindet sich noch eine graphische Darstellung der Einzelverbräuche nach Komponenten (aber für alle Nutzungsprofile gemeinsam auf das Jahr betrachtet, wie in der oberen Übersichtsdarstellung) als Sankey-Diagramm. Die Dicke der einzelnen Stränge ist in dieser Darstellung proportional zum entsprechenden Verbrauch des Aggregats bzw. Teilprozesses.

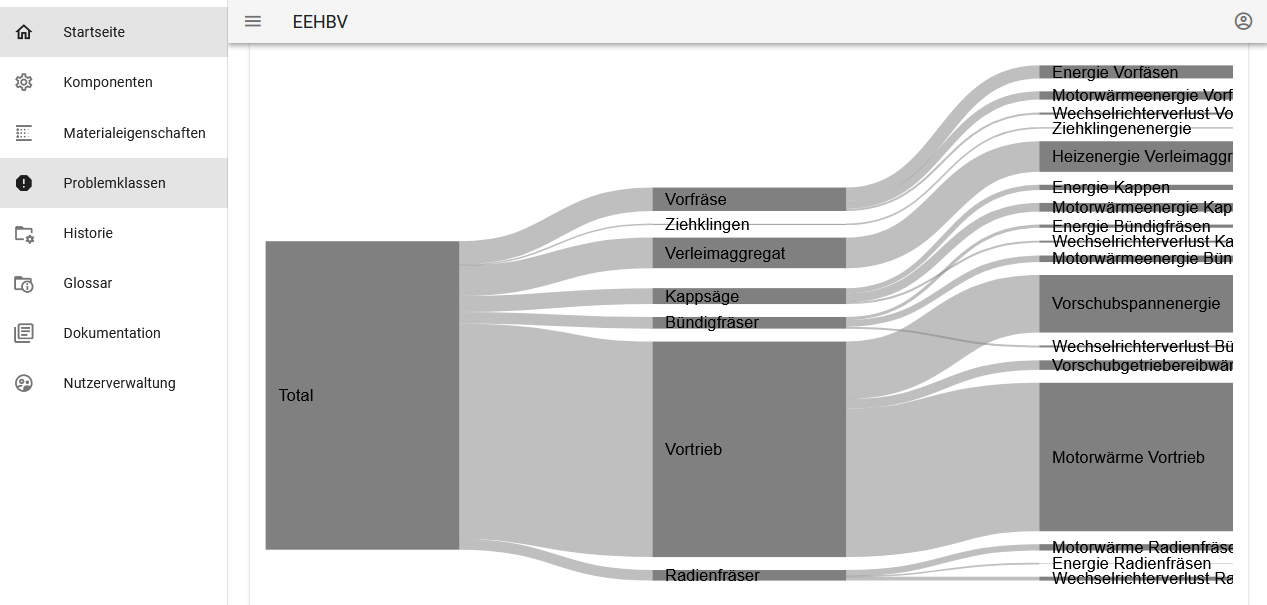


Abbildung 16: Darstellung der Teilverbräuche der einzelnen Prozesse bzw. Komponenten in der Ergebnisansicht als Sankey-Diagramm.

Durch Anklicken einer Vorschlag-Zeile bzw. auch Variante kann ausgewählt werden, für welchen konkreten Komponentenvorschlag die Tabelle und das Sankey-Diagramm angezeigt werden sollen.

## Betrachten von bereits ausgeführten Optimierungsanfragen

Ausgeführte Optimierungsanfragen werden mit Ergebnis in der Datenbank abgespeichert. Über den Hauptmenüpunkt Historie kann man zunächst eine Problemklasse auswählen, für die man bereits ausgeführte Optimierungen anzeigen lassen möchte und die Klasse über den ANZEIGEN-Button bestätigen. Aus der dann angezeigten Liste von Optimierungsanfragen kann man die gewünschte durch Anklicken der Spalte anzeigen lassen.

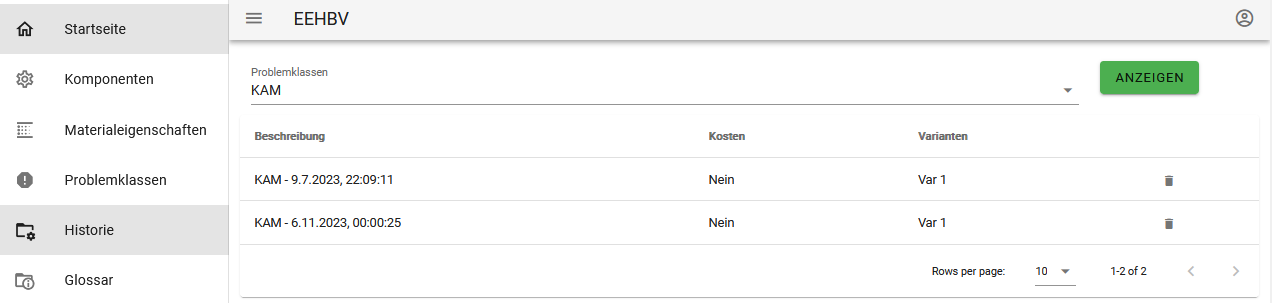


Abbildung 17: Die Anzeige bereits durchgeführter Optimierungen erfolgt über den Menüpunkt Historie und die Auswahl einer Problemklasse.

## Anlegen/Bearbeiten/Importieren von Komponentendaten

Die für Nutzer ab der Nutzerrolle data verfügbare Funktion zum Anlegen neuer Komponenteneinträge (NEUER EINTRAG) befindet sich über der Übersichtstabelle auf der jeweiligen Komponentenansicht.

Einzelne bestehende Einträge können über die Stift- und Mülleimer-Symbole in der letzten Spalte (Aktionen) der Tabelle geändert bzw. gelöscht werden. Vor dem Löschen findet eine Sicherheitsabfrage statt.

Über das Datei-Symbol neben der Funktion zum Anlegen neuer Einträge können auch mehrere Einträge gleichzeitig als Excel- oder CSV-Datei importiert werden. In der Datei muss die erste Zeile als Spaltenköpfe die Namen der **Datenbank**spalten des Komponententyps enthalten. Diese können über einen Klick auf das Tabellensymbol über der Tabelle (links neben Import) angezeigt werden. In CSV-Dateien müssen die Spalteneinträge durch Kommata separiert sein und Dezimalstellen müssen durch einen Punkt angegeben werden.

## Anlegen von Komponententypen

Tabellenname: Konvention component\_typeinenglishplural, API-Name: englishplural

## Anlegen/Bearbeiten von Materialeigenschaften

## Anlegen/Bearbeiten von Glossareinträgen

## Anlegen/Bearbeiten von Nutzern

## Definition neuer Problemklassen

Vor der Anlage neuer Problemklassen sollten Sie sich eingehend mit dem Vorgehensrahmenmodell beschäftigt und Ihr Problem ausreichend analysiert haben. Auch sollte die Anlage der für Ihre Problemklasse notwendigen Komponententypen und Materialeigenschaftstypen vor der eigentlichen Problemdefinition erfolgen. Der gesamte Prozess dabei inklusive der Anlage in der Software ist im Abschnitt Vorgehensrahmenmodell beschrieben.

# Vorgehensrahmenmodell

In diesem Abschnitt wird das zugrundeliegende Vorgehensmodell vorgestellt, mit dem Prozesse analysiert werden können, um als Optimierungsproblem in der Software beschrieben werden zu können. Zusammen mit der Nutzung des Programmes dient dieses Vorgehensmodell als Framework zur Lösung einer Klasse von Problemen. Erläutert wird dieses Vorgehensmodell am Beispiel von Kantenanleimmaschinen. Diese dienen in der Möbelfertigung dem Anbringen von Furnieren an den Schmalflächen von Brettern. Dieser Maschinentyp wurde im Rahmen des Forschungsprojekts, in dem auch diese Software entwickelt wurde, eingehend analysiert.

Das Vorgehensmodell gliedert sich in folgende Teilschritte:

1. Identifikation der Komponententypen/Aggregate, aus denen das System zusammengesetzt werden kann
2. Identifikation und Systematisierung der Varianten, in denen das System aufgebaut werden kann
3. Modellierung der Prozess- und Verlustleistungen der einzelnen Prozessschritte als Funktion
4. Auflistung der prozess- und komponentenabhängigen Variablen
5. Identifikation impliziter Nebenbedingungen
6. Entwurf Abarbeitungs-/Auswahlalgorithmus
7. Datenbankmodell für Komponenten anlegen und mit Daten befüllen
8. Materialeigenschaften identifizieren, die als Prozessparameter relevant sind, und Datenbank befüllen
9. Anlegen des Prozessmodells in der Anwendung als Problemklasse

Die in den Schritten 1 bis 6 erhobenen Informationen werden im Schritt 9 menügeführt abgefragt und eine neue Problemklasse angelegt, an die anwenderspezifische Optimierungsanfragen gestellt werden können.

## Identifikation der Komponententypen/Aggregate

Die betrachtete Kantenanleimmaschine besteht (vereinfacht betrachtet) aus mehreren Fräsaggregaten, die jeweils aus einem Frequenzumrichter, Motor, evtl. pneumatischen Stellkolben zur Mitführung am Werkstück und dem eigentlichen Werkzeug bestehen, dem Leimaggregat (Heizelement), einem Kappaggregat (Pneumatik, Motor, Frequenzumrichter) zum Abtrennen des von einer Rolle zugeführten Kantenbandes, Ziehklingen zum Entfernen von Kleberresten, und dem Nebenantrieb (Motor, Getriebe, Förderband).

## Identifikation und Systematisierung der Varianten

Für die Analyse der Varianten wurde der Maschinenkatalog eines deutschen Herstellers von Kantenanleimmaschinen auf die verbauten Komponenten der verschiedenen Modelle hin analysiert (Tabelle 1). Je nach Anwendungsfall des Kunden sind verschiedene Bearbeitungsaggregate notwendig, was sich auch in der Vielzahl der angebotenen Modelle widerspiegelt. Im konkreten Anwendungsfall beim Entwurf einer neuen Maschine ist vom Endnutzer abzufragen, welche Varianten für ihn in Frage kommen. Dafür bietet das Programm eine Fragensystematik, die entweder eine feste Liste oder die Form eines Fragenbaumes haben kann (die vorhergehenden Antworten bestimmen die weiteren gestellten Fragen). Die baumförmige Fragesystematik sollte bei jeder letzten Frage zu einer eindeutigen Auswahl der gewünschten Variante führen. Auch bei einer Liste sollten die Fragen so gewählt sein, dass die Beantwortung aller Fragen die Varianten möglichst einschränken. Bei der Anfrage kann man dann auch immer alternativ die gewünschten zu betrachtenden Varianten aus einer Liste ankreuzen.

Tabelle 1: Modellvielfalt eines Herstellers von Kantenanleimmaschinen hinsichtlich unterschiedlicher verbauter Aggregate

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Fügefräser** | **Verleimaggregat** | **NIR** | **Heißluft** | **Kappaggregat** | **Bündigfräsen** | **Profilfräsen** | **Eckenkopieren** | **Multifunktions-Formfräsaggregat** | **Profilziehklinge** | **Leimfugenziehklinge** | **Schwabbelaggregat** | **Eckschwabbelaggregat** | **Nutfräsaggregat** | **Universalfräsaggregat** |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 2 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 2 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 2 | 2 | 2 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 2 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 2 |
| 1 | 1 | 0 | 0 | 1 | 2 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 2 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 2 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 2 | 0 | 1 | 2 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 2 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 2 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 2 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 2 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 2 | 0 | 0 | 0 |

Zu Testzwecken wurde diese Aggregatmatrix leicht reduziert und ein Fragebaum entworfen, der der Einfachheit halber nur abfragt, ob ein Aggregat enthalten sein soll oder nicht. Dabei wurde die Reihenfolge der abgefragten Aggregate so gewählt, dass möglichst wenige Fragen gestellt werden müssen, aber alle im Katalog identifizierten Varianten abgedeckt werden. In Abbildung 18 ist dieser Fragebaum mit den jeweils noch zur Verfügung stehenden Varianten nach Beantwortung einer Frage abgebildet, wie er im Programm bei der Anlage des Fragebaums dargestellt wird.



Abbildung 18: Fragebaum zur Auswahl der Varianten aus der in Tabelle 1 dargestellten Variantenvielfalt, wie er im Programm bei der Anlage dargestellt wird. Die Daumen hoch oder runter an den Knoten im Baum bedeuten, dass die vorige Frage positiv bzw. negativ beantwortet wurde. An jedem Knoten befindet sich eine Liste der nach der Antwort noch zur Verfügung stehenden Varianten und eine weitere Frage, falls noch mehrere Varianten zur Verfügung stehen.

Für die finale Anlage des Modells wurde auf die Variantenvielfalt verzichtet, da für jede Variante jeweils ein Teilmodell hätte hinterlegt werden müssen und nicht alle Aggregattypen für Tests zur Verfügung standen.

## Modellierung der Prozess- und Verlustleistungen der einzelnen Prozessschritte als Funktion

Jede am System abfallende Leistung muss numerisch als Funktion in Python-Code dargestellt werden. Jede dieser Funktionen sollte als eigenes Modul in einer Datei vorbereitet werden. Jedes Modul muss eine Funktion mit dem Namen target\_func enthalten, die die gewünschten Werte als Skalar oder Liste zurückliefert. Das Modul kann weitere Funktionen oder auch Klassen enthalten, die von dieser Haupt-Funktion genutzt werden. Es können auch Bibliotheken importiert werden. Neben den Standard-Bibliotheken stehen für Berechnungen pandas und numpy zur Verfügung. Weitere Bibliotheken können hinzugefügt und ebenfalls im dynamischen Code genutzt werden. Dafür muss allerdings die Auslieferungsumgebung der Anwendung angepasst werden.

Bei der Anlage des Modells im Programm wird das Berechnungsergebnis einer Funktion dann einer Variablen zugeordnet (Namenskonvention Kleinschreibung, beginnend mit l\_), die in später aufgerufenen Funktionen verwendet werden kann. Auch kann angegeben werden, ob der Rückgabewert in die Energiebetrachtung eingehen soll oder ein Zwischenergebnis ist.

Die analytischen Leistungsmodelle wurden für die Integration in die Software in numerische Funktionen überführt. Dabei wurden Last- und Leerlaufzeiten vereinfachend als zwei stationäre Zustände angenommen. Dynamische Verlaufskurven könnten aber auch durch entsprechende Parametrisierung abgebildet werden. Eventuellen kurzzeitigen Drehmomentspitzen zum Beginn von Werkzeugeingriffen, die zur Überschreitung von Randbedingungen führen könnten, kann dadurch Rechnung getragen werden, dass ein Sicherheitsfaktor in den Bedingungen entsprechend ausgelegt wird. Auf pneumatische Prozesse wurde in der folgenden Darstellung verzichtet, um die Komplexität des Beispiels zu reduzieren. Um auf einzelne Elemente von Zwischenergebnissen in Listenform zuzugreifen, enthält das Modell ein Hilfsmodul, das nur für diesen Zugriff gedacht ist. Eine Auflistung aller Python-Funktionen für das Beispiel findet sich in Unterabschnitt Funktionen für Beispiel-Prozess.

## Auflistung der Variablen

Die Eingangsvariablen der einzelnen Leistungsfunktionen werden dahingehend analysiert, ob es sich um Eigenschaften der verwendeten Komponenten handelt, deren konkrete Ausprägung dann Objekt der Optimierung sein wird, oder um Betriebsparameter, die bei der Abfrage vom Anwender als Nutzungsprofil für den Prozess abgefragt werden müssen (z. B. die gewünschte Vorschubgeschwindigkeit des Nebenantriebs). Bei der der Eingabe des Modells in die Software werden für die Parameter bestimmte Namenskonventionen erwartet (Kleinschreibung, Komponentenparameter beginnend mit c\_, Betriebsparameter beginnend mit p\_). Daher sollte schon bei der Aufstellung der Parameter im Vorfeld auf diese Konventionen geachtet werden. Die Betriebsparameter können auch als Konstanten mit einem für den Prozess sinnvollen Wert angenommen werden. So werden im Beispiel die geometrischen Parameter des Klebstoffbehälters als konstant angenommen. Die vom Nutzer einzugebenden Parameter können auch Materialkonstanten sein, im Fall der Kantenanleimmaschine z. B. die spezifische Schnittkraftkonstante der verarbeiteten Werkstücke. Diese Parameter können bei der Anlage des Prozesses in der Anwendung mit zuvor in der Datenbank angelegten Eigenschaften verknüpft werden und bei der Anfrage dann ein Datenbankeintrag aus einem Dropdown-Menü ausgewählt werden.

Neben den bei einer Optimierungsanfrage vom Nutzer einzugebenden Parametern werden auch die Berechnungsergebnisse der Funktionen neuen Variablen zugewiesen, die als Eingangsparameter für weitere Funktionen dienen können. In Tabelle 2 sind alle im Modell für die Kantenanleimmaschine verwendeten Parameter im Kontext ihres Aufrufs aufgelistet. Aus dieser Tabelle wird auch ersichtlich, dass Funktionen mehrfach mit unterschiedlichen Parametern aufgerufen werden können. Bei der Anlage des Modells in der Anwendung wird man also neben den Funktionen und Parametern eine Liste dieser Aufrufe in einer festzulegenden Reihenfolge angeben müssen.

## Identifikation impliziter Nebenbedingungen

Für die Optimierung ist wichtig, die impliziten Bedingungen zu erkennen, die sich aus dem Nutzungsprofil ergeben, um geeignete Komponenten auswählen zu können. So muss der Motor bzw. die Kombination aus Motor und Getriebe des Nebenantriebs so gewählt sein, dass die gewünschte Vorschubgeschwindigkeit erreicht werden kann oder die Antriebsmotoren der Fräsaggregate müssen ausreichend Leistung haben, um die Prozessleistung am Werkstück erbringen zu können. Das bedeutet, dass die Nebenbedingungen als Anforderungen in den Parametern definiert werden müssen und es muss identifiziert werden, auf welche Komponentenauswahl im Optimierungsprozess die Nebenbedingung abgeglichen werden muss, um Komponentenkonstellationen auszuschließen, die die Anforderungen nicht erfüllen. Zu diesem Schritt gehört auch die Festlegung der Reihenfolge, in der der Optimierungsalgorithmus die Komponenten auszuwählen und die jeweiligen Verlust- bzw. Prozessleistungen zu berechnen hat. Dabei muss berücksichtigt werden, dass das Ergebnis einiger Funktionen als Input für andere dient oder die Auswahl der nächsten Komponente einschränkt und diese entsprechend kausal geordnet werden müssen.

In Tabelle 2 sind auch die Randbedingungen aufgelistet, die sich aus entsprechenden Überlegungen zu den einzelnen Komponenten ergeben.

Tabelle 2: Auflistung aller Parameter, Funktionsaufrufe und Randbedingungen mit Zuordnung zur jeweiligen Komponente, an der die entsprechende Leistung abfällt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Aggregat** | **Funktionen** | **Rand-bedingungen** | **Komponenten** | **Kompo-nenten-parameter** | **Prozess-parameter** |
|  |  |  |  |  | Werkstückdicke  p\_width  Werkstücklänge  p\_length  Breite Kantenband  p\_width\_band  Dicke Kantenband  p\_thickness\_band  Werkstücklücke  p\_gap  Abnutzungsfaktor für Fräsen  p\_wear\_factor  Vorschubgeschwindigkeit  p\_feed\_speed  spez. Schnittkraftkonstante Werkstück  p\_kc05  spez. Schnittkraftkonstante Kantenband  p\_kc05\_band |
|  | Überlapp overlap.py  Ausgabe  l\_overlap\_band  Eingabe  p\_width, p\_width\_band |  |  |  |  |
| Verleimen | Heizleistung Verleimaggregat  glue.py  Ausgabe  l\_glue\_pow  Eingabe: alle Prozessparameter des Aggregats |  |  |  | Spez. Wärmeleitfähigkeit Klebstoffbehälter  p\_glue\_conductivity  Oberfläche Klebstoffbehälter  p\_glue\_surface  Wanddicke Klebstoffbehälter  p\_glue\_d |
| Fügefräsen |  |  | Fräswerkzeug, wechselbar, abgebildet durch Prozessparameter |  |  |
|  | Fräsleistung Vorfräsen process\_power\_rotating.py  Listenselektion  select\_from\_ist.py  Ausgabe:  l\_pre\_pow  l\_pre\_pow\_max  l\_pre\_m  Eingabe: alle Prozessparameter des Aggregats,  p\_width, p\_length, p\_gap, p\_feed\_speed, p\_kc05 |  |  |  | Zahnzahl  p\_pre\_teeth (3)  Frästiefe  p\_pre\_depth 3 mm  Schnittwinkel p\_pre\_ang\_kappa 90 °  p\_pre\_ang\_lambda 65 °  Drehzahl  p\_pre\_n (9000/min)  Werkzeugdurchmesser  p\_pre\_diameter 125mm |
|  |  |  | Vorfräsmotor  c\_pre\_motor | Nenndrehzahl n\_nominal  Nennleistung p\_nominal  Nenndrehmoment m\_nominal  Koeff. A-G coeff\_a … coeff\_g |  |
|  |  | 0 <= c\_motor\_pre.p\_nominal - p\_wear\_factor  x l\_pre\_pow\_max |  |  |  |
|  |  | c\_motor\_pre.f\_nominal \* 60 / c\_motor\_pre.n\_pole\_pairs > p\_pre\_n |  |  |  |
|  |  | c\_motor\_pre.m\_nominal > l\_pre\_m |  |  |  |
|  | Motorverlust Vorfräsen  loss\_motor.py  Ausgabe  l\_pre\_motor\_loss  Eingabe  p\_length, p\_gap, p\_feed\_speed, p\_pre\_n, l\_pre\_m |  |  |  |  |
|  |  |  | Vorfräsumrichter  c\_pre\_converter | Messpunkte P(x;y) |  |
|  |  | 0 >= abs(c\_pre\_converter.p\_nominal - c\_pre\_motor.p\_nominal) / c\_pre\_converter.p\_nominal - 10 |  |  |  |
|  |  | 0 <= c\_pre\_converter.n\_nominal x 60 - p\_pre\_n |  |  |  |
|  | Wechselrichterverlust Vorfräsen  Ausgabe  l\_pre\_converter\_loss  Eingabe  Messpunkte, p\_length, p\_gap, p\_pre\_n, l\_pre\_m, c\_pre\_motor.m\_nominal, c\_pre\_motor.n\_nominal |  |  |  |  |
| (Verleimteil) |  |  |  |  |  |
| Kappen |  |  | Schneidewerkzeug, wechselbar(?),  abgebildet durch Prozessparameter |  |  |
|  | Abwärtsgeschwindigkeit, Last-Leerlauf-Verhältnis Kappen  trim\_values.py  Listenselektion  Ausgabe  l\_trim\_speed  l\_trim\_t\_load, l\_trim\_t\_idle  Eingabe  p\_feed\_speed, p\_trim\_angle, p\_width  Fräsleistung Kappen  Listenselektion  Ausgabe  l\_trim\_pow  l\_trim\_pow\_max  l\_trim\_m  Eingabe:  Alle Prozessparameter des Aggregats  l\_overlap\_band, p\_thickness, l\_trim\_speed, p\_kc05\_edge, l\_trim\_t\_load, l\_trim\_t\_idle  Beachten: 2 \* Leistung!,  Schneidendicke als Breite, Kantenbanddicke als Frästiefe |  |  |  | Zahnzahl  p\_trim\_teeth (24)  Schnittwinkel p\_trim\_ang\_kappa 65 °  p\_trim\_ang\_lambda 5 °  Drehzahl  p\_trim\_n (12000 / min)  Werkzeugdurchmesser  p\_trim\_diameter 120 mm  Schneidendicke  p\_trim\_width 3.2 mm  Winkel Führung  p\_trim\_angle 45 ° |
|  |  |  | Kappmotor  c\_trim\_motor |  |  |
|  |  | c\_motor\_trim.p\_nominal > p\_wear\_factor  x l\_trim\_pow\_max |  |  |  |
|  |  | c\_motor\_trim.f\_nominal \* 60 / c\_motor\_trim.n\_pole\_pairs > p\_trim\_n |  |  |  |
|  |  | c\_motor\_trim.m\_nominal > l\_trim\_m |  |  |  |
|  | Motorverlust Kappen  Ausgabe  l\_trim\_motor\_loss  Eingabe  l\_trim\_t\_load, l\_trim\_t\_idle  Beachten: 2\* Leistung |  |  |  |  |
|  |  |  | Kappumrichter  c\_trim\_converter |  |  |
|  |  | 0 >= abs(c\_trim\_converter.p\_nominal - c\_trim\_motor.p\_nominal) c\_trim\_converter.p\_nominal - 10 |  |  |  |
|  |  | 0 <= c\_trim\_converter.n\_nominal x 60 - p\_trim\_n |  |  |  |
|  | Wechselrichterverlust Kappen  Ausgabe  l\_trim\_converter\_loss  Beachten: 2 \* Leistung!  Eingabe  Messpunkte, l\_trim\_t\_load, l\_trim\_t\_idle, p\_trim\_n, l\_trim\_m, c\_trim\_motor.n\_nominal, c\_trim\_motor.m\_nominal |  |  |  |  |
| Bündig-fräsen |  |  |  |  |  |
|  | Fräsleistung Bündigfräsen  Ausgabe  l\_flush\_pow  l\_flush\_pow\_max  l\_flush\_m  Eingabe:  Alle Prozessparameter des Aggregats  l\_overlap\_band, p\_thickness\_band  Beachten: 2 \* Leistung!, ½ Overlap als Breite |  |  |  | Zahnzahl  p\_flush\_teeth (3)  Schnittwinkel p\_flush\_ang\_kappa 90 °  p\_flush\_ang\_lambda 65 °  Drehzahl  p\_flush\_n (12000 / min)  Werkzeugdurchmesser  p\_flush\_diameter 80 mm |
|  |  |  | Bündigfräsmotor  c\_flush\_motor |  |  |
|  |  | c\_motor\_flush.p\_nominal > p\_wear\_factor  x l\_flush\_pow\_max |  |  |  |
|  |  | c\_motor\_flush.f\_nominal \* 60 / c\_motor\_flush.n\_pole\_pairs > p\_flush\_n |  |  |  |
|  |  | c\_motor\_flush.m\_nominal > l\_flush\_m |  |  |  |
|  | Motorverlust Bündigfräsen  Ausgabe  l\_flush\_motor\_loss  Beachten: Leistung \* 2!  Eingabe  p\_length, p\_gap, p\_flush\_n, l\_flush\_m |  |  |  |  |
|  |  |  | Bündigfräsumrichter  c\_flush\_converter |  |  |
|  |  | 0 >= abs(c\_flush\_converter.p\_nominal - c\_flush\_motor.p\_nominal) / c\_flush\_converter.p\_nominal - 10 |  |  |  |
|  |  | 0 <= c\_flush\_converter.n\_nominal x 60 - p\_flush\_n |  |  |  |
|  | Wechselrichterverlust Bündigfräsen  Ausgabe  l\_flush\_conv\_loss  Beachten: 2 \* Leistung!  Eingabe  Messpunkte, p\_flush\_n, l\_flush\_m, c\_flush\_motor.n\_nominal, c\_flush\_motor.m\_nominal |  |  |  |  |
| Radien-fräsen |  |  |  |  |  |
|  | Fräsleistung Radienfräsen  Listenselektion  Ausgabe  l\_round\_pow  l\_round\_pow\_max  l\_round\_m  Eingabe:  alle Prozessparameter des Aggregats  p\_length, p\_gap, p\_feed\_speed  Beachten: 2 \* Leistung!, ½ Overlap als Breite |  |  |  | Zahnzahl  p\_round\_teeth (4)  Fräsradius  p\_round\_radius  Drehzahl  p\_round\_n (8700 / min)  Werkzeugdurchmesser  p\_round\_diameter 125 mm  Winkel lambda ??? |
|  |  |  | Radienfräsmotor  c\_round\_motor |  |  |
|  |  | c\_motor\_round.p\_nominal > p\_wear\_factor  x l\_round\_pow\_max |  |  |  |
|  |  | c\_motor\_round.f\_nominal \* 60 / c\_motor\_round.n\_pole\_pairs > p\_round\_n |  |  |  |
|  |  | c\_motor\_round.m\_nominal > l\_round\_m |  |  |  |
|  | Motorverlust Radienfräsen  Ausgabe  l\_round\_motor\_loss  Eingabe  p\_length, p\_gap, p\_round\_n, l\_round\_m |  |  |  |  |
|  |  |  | Radienfräsumrichter  c\_round\_converter |  |  |
|  |  | 0 >= abs(c\_round\_converter.p\_nominal - c\_round\_motor.p\_nominal) / c\_round\_converter.p\_nominal - 10 |  |  |  |
|  |  | 0 <= c\_round\_converter.n\_nominal x 60 - p\_round\_n |  |  |  |
|  | Wechselrichterverlust Radienfräsen  Ausgabe  l\_round\_conv\_loss  Eingabe  Messpunkte, p\_length, p\_gap, p\_round\_n, l\_round\_m, c\_round\_motor.f\_nominal, c\_round\_motor.m\_nominal, c\_round\_motor.p\_nominal |  |  |  |  |
| Ziehklingen |  |  |  |  |  |
|  | process\_power\_scraper.py  Ziehklinkenleistung  process\_power\_scraper.py  Ausgabe  d\_scraper\_pow  Eingabe  alle Prozessparameter des Aggregats, p\_feed\_speed, p\_width, p\_length, p\_gap |  |  |  | Einstellwinkel  p\_scraper\_ang\_kappa 90°  Eingriff  p\_scraper\_depth 0.0002m |
| Neben-antrieb |  |  | Band/Rollen/Federn |  | Spannfederpunkte  p\_feed\_x1,  p\_feed\_y1,  p\_feed\_x3,  p\_feed\_y3  Federkonstante  p\_feed\_kc 49660 N/m  Reibungskoeffizient  p\_feed\_mu 6  Transferbandlänge  p\_feed\_l  Abstand Federelement  p\_feed\_spring\_dist 0.07 m  Spannweg Werkstück  p\_feed\_delta\_x  Vorspannung Feder  p\_feed\_f\_zero  Durchmesser Riemenscheibe  p\_feed\_disc\_diam 0.2 m |
|  | Spannleistung, -moment  Ausgabe  l\_feed\_pow,  l\_feed\_m  Eingabe  p\_feed\_spring\_dist, p\_feed\_mu, p\_feed\_kc, p\_feed\_f\_zero, p\_feed\_speed, p\_feed\_x1,  p\_feed\_y1,  p\_feed\_x3,  p\_feed\_y3,  p\_length, p\_gap, i\_gears, disc\_diam |  |  |  |  |
|  |  |  | Getriebe Bandvorschub  c\_feed\_gears | Getriebewirkungsgrad  efficiency  Getriebeübersetzung  gear\_ratio  Getriebenennmoment  m\_mominal |  |
|  | Getriebewerte  Ausgabe  l\_feed\_gear\_loss, l\_feed\_m\_total, l\_feed\_omega  Eingabe |  |  |  |  |
|  |  | c\_feed\_gears.m\_nominal >= l\_feed\_m |  |  |  |
|  |  |  | Bandvorschubmotor  c\_feed\_motor |  |  |
|  |  | c\_motor\_feed.p\_nominal >  l\_feed\_pow + l\_feed\_gear\_loss |  |  |  |
|  |  | c\_motor\_feed.f\_nominal \* 60 / c\_motor\_feed.n\_pole\_pairs > l\_feed\_f\_in |  |  |  |
|  |  | c\_motor\_feed.m\_nominal > l\_feed\_m\_total |  |  |  |
|  | Motorverlust Vorschub  Ausgabe  l\_feed\_motor\_loss  Eingabe  p\_length, p\_gap, l\_feed\_omega, l\_feed\_m\_total |  |  |  |  |

## Entwurf Auswahlalgorithmus

Ein Optimierungsalgorithmus, der für alle Prozesse die optimalen Ergebnisse liefert, ist bereits in das System integriert. Für spezielle Problemklassen könnten aber andere Optimierungsansätze zu effizienteren Algorithmen führen. Auch der Optimierungsalgorithmus selbst ist Teil des Prozessmodells und kann in der Datenbank gespeichert werden und wird dann bei Anfragen dynamisch geladen.

Wie für die Leistungsfunktionen gilt, dass der Optimierungscode als eigenes Modul bereitgestellt werden kann, dass auch externe Bibliotheken nutzen kann. Einstiegspunkt für das Modul muss eine Funktion mit dem Namen call\_solver sein, der drei Variablen übergeben werden. An den ersten Parameter werden die dynamisch geladenen Leistungsfunktionen übergeben, die für die Problemklasse definiert wurden. Der zweite Parameter enthält das restliche Modell mit Definition der Aufrufe und ihrer Reihenfolge und die Randbedingungen. Der dritte Parameter enthält die Daten aller Komponententypen, die für die Problemklasse relevant sind. Diese wurden vor dem Aufruf des Optimierungscodes aus der Datenbank geladen und dienen in dem Modul als Datengrundlage, auf der die Optimierung ausgeführt wird. Die Rückgabewerte der Funktion sind die n energetisch besten Komponentenzusammenstellungen, die sich aus den eingegebenen Nutzeranforderungen ergeben, eine optionale Kostenbetrachtung und eventuelle Hinweistexte, falls die Berechnung mit den eingegebenen Parametern nicht möglich ist.

## Datenbankmodell für Komponenten anlegen und mit Daten befüllen

Für alle Komponententypen muss ein Modell in der Datenbank über das Komponentenmenü angelegt werden. Als vorgegebene Spalten werden der Komponenten- und Herstellername als Identifikationsmerkmale und der Preis (Wert optional anzugeben) für die Ermöglichung einer Amortisationsrechnung angezeigt, die weiteren notwendigen Eigenschaften des Komponententyps können frei mit den jeweiligen Einheiten angegeben werden. Als Datentypen stehen dabei numerische Werte, Wahrheitswerte und Zeichenketten zur Verfügung (Abbildung 14). Die wesentlichen Komponententypen für die Kantenanleimmaschine sind Motoren und Frequenzumrichter. Weiterhin sind für den Vorschubmotor Getriebedaten notwendig. Die Fräswerkzeuge wären prinzipiell auch als Komponentenmodell umsetzbar, wurden für den Anwendungsfall aber als vorgegebene Standardwerte implementiert. Auf die Betrachtung verschiedener Heizaggregate bzw. auch anderer Module für das Leimaggregat (z. B. Laser oder Heißluft) wurden im Rahmen dieser Analyse verzichtet. Die Daten einzelner zur Verfügung stehender Komponenten können händisch in die angelegten Modelle eingetragen oder als Liste aus einer CSV-Datei importiert werden.

## Materialeigenschaften in Datenbank anlegen

Prozessparameter, die sich aus Materialeigenschaften ergeben, können ebenfalls in der Datenbank in einer eigenen Tabelle angelegt werden. Für unser Anwendungsbeispiel ist die empirisch bestimmte Schnittkraftkonstante der Werkstücke und Kantenbänder relevant, die in das Schnittleistungsmodell eingeht. Da dieser Wert für Holzarten mit einiger Schwankung angegeben wird, sollten aus entsprechender Literatur die Werte am oberen Rand des angegebenen Bereichs gewählt werden.

## Anlegen des Prozessmodells in der Anwendung als Problemklasse

Erst nach Vorbereitung durch die vorhergehenden Schritte sollte mit der Eingabe in die Anwendung begonnen werden. Die Eingabe des Modells erfolgt in sechs übergeordneten Schritten.

Im ersten Schritt (Abbildung 19) werden generelle Informationen wie der Name des Maschinentyps und ein für die interne Handhabung durch das Programm notwendiger Pfadname für Aufrufe des Prozesstyps. Da dieser Pfad Teil einer URL ist, sollte er keine Leer- und Sonderzeichen oder Umlaute enthalten. In diesem Menü wird auch angegeben, ob man sich bei der Variantenauswahl für eine Listen- oder Baumabfrage entschieden hat.

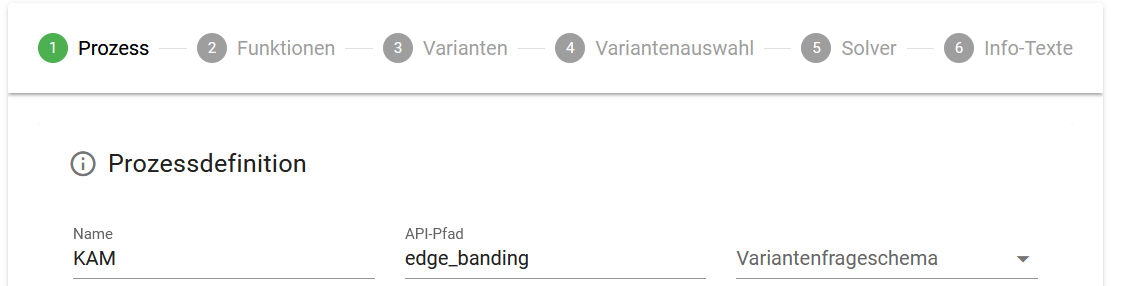


Abbildung 19: Erster Reiter der menügeführten Eingabe von Prozessmodellen. Neben einem Anzeigenamen und einem internen Namen ist das Frageschema für die Variantenauswahl anzugeben.

Bei der auf der gleichen Seite vorzunehmenden Eingabe von Prozessparametern (Abbildung 20) sind ein Anzeigename und der Variablenname für die Weitergabe in der Optimierung anzugeben. Die Einheit ist optional, da sie an keiner Stelle auf Konsistenz geprüft wird, sollte aber für die Übergabe an die Leistungsfunktionen angepasst sein (bzw. in den Funktionen entsprechende Konvertierungen vorgenommen werden). Neben der Angabe von Standardwerten ist die Zuweisung der Variable zu einer zuvor angelegten Materialkonstante möglich. In der Abfrage der Prozessparameter bei einer Optimierungsanfrage wird dann die Liste der entsprechend eingetragenen Werkstoff-Werte als Drop-Down angezeigt. Als allgemein ist ein Parameter zu kennzeichnen, wenn er für alle Anwendungsszenarien einer zu planenden Maschine den gleichen Wert haben wird, z. B. weil er eine feste geometrische Größe der Maschine angibt.

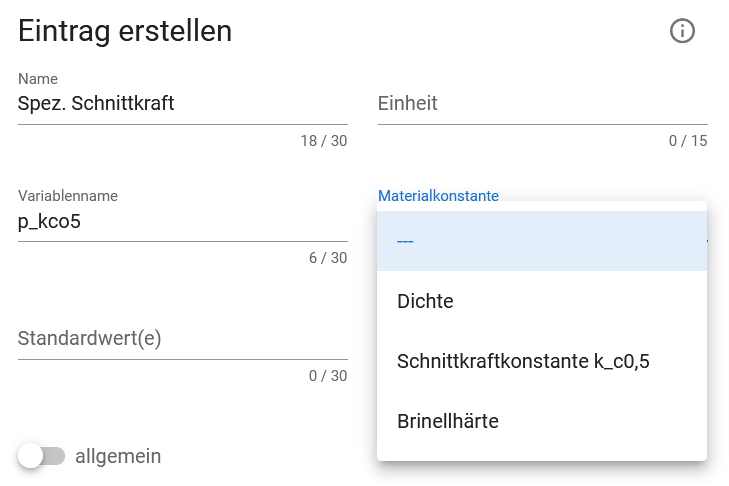


Abbildung 20: Formular zur Angabe von Prozessparametern. Es kann ein Standardwert angegeben und die Zuordnung zu einer Materialkonstante vorgenommen werden.

Im nächsten Reiter werden die Funktionen eingegeben. Neben der Beschreibung für eine spätere Referenzierung und der Funktion selbst als Python-Code (Python-spezifische Formatierung beachten, am besten aus vorbereiteten Dateien in das Formularfeld kopieren) kann eine Dokumentation angegeben werden, die bei der in folgenden Schritten durchzuführenden Definition von Funktionsaufrufen hilfreich ist.

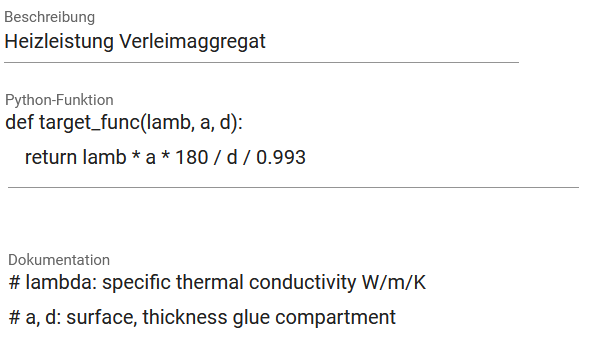


Abbildung 21: Formular für die Eingabe von Funktionen, die optionale Dokumentation wird später bei der Definition von konkreten Aufrufen der Funktion als Hilfe angezeigt.

Im dritten Reiter erfolgt die Eingabe von Varianten, in denen die Maschine anlegbar sein soll. Eine Variante besteht neben ihrem Namen aus Komponenten (Abbildung 22). Einer Komponente muss ein zuvor angelegter Typ zugewiesen werden. Der Variablenname wird beim Aufruf von Leistungsfunktionen während der Optimierung intern weiterverwendet. Die Zuordnung zu einem Aggregat gibt eine Gruppierung von Komponenten an, die von einander abhängige Funktionen und Nebenbedingungen haben. Es können in diesem Feld neue Aggregatnamen eingegeben oder aus der Liste bereits bestehender ausgewählt werden. Die Komponenten müssen in der Reihenfolge angegeben werden, in der sie im Optimierungsalgorithmus ausgewählt werden, weil das Ergebnis der Leistungsfunktion der zuerst angegebenen Komponente als Parameter in Bedingungen oder Leistungsfunktionen von später eingegebenen Komponenten eingehen. Die Reihenfolge kann bei der Eingabe durch die in der Abbildung im Hintergrund unten rechts erkennbaren Pfeile nachträglich geändert werden.

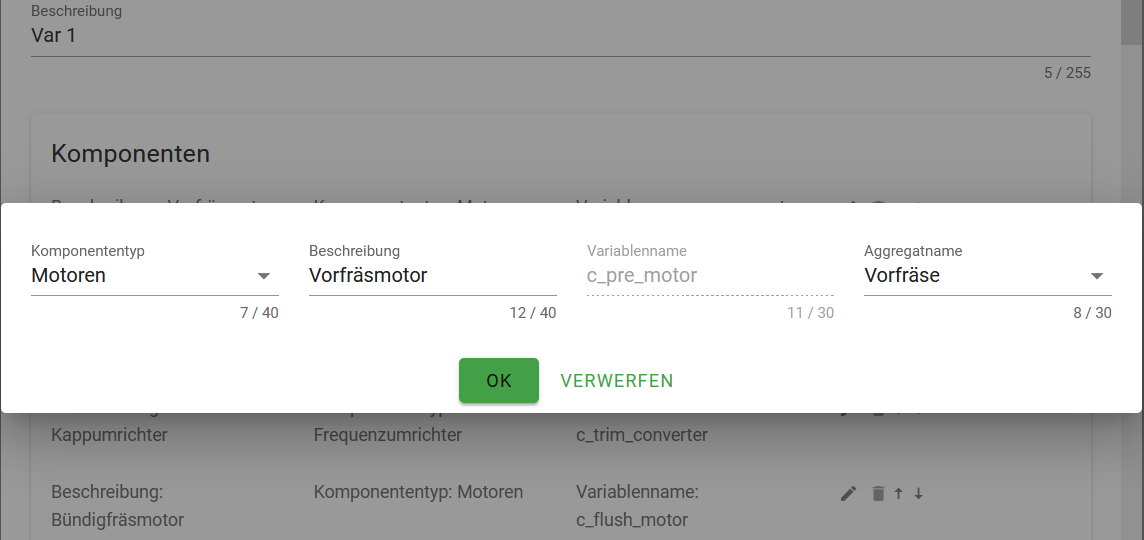


Abbildung 22: Formular zur Anlage von Varianten-Komponenten; die Auswahl des Typs erfolgt per Drop-Down aus den zuvor in der Datenbank angelegten Komponententypen.

Nach der Anlage der Komponenten können im gleichen Reiter weiter unten die Funktionsaufrufe für die Variante definiert werden (Abbildung 23). Die anzugebende Beschreibung ist vor allem für eine nachträgliche Bearbeitung des Modells hilfreich. Es kann angegeben werden, nach der Auswahl welcher Komponente die Funktion auszuwerten ist und zu welchem Aggregat der Funktionsaufruf zuzuordnen ist. Der Schieberegler Zielfunktion gibt an, ob der Rückgabewert der Funktion in den Gesamtenergieverbrauch bei der Optimierung eingehen soll (ausgewählt) oder Zwischenwert für weitere Berechnungen ist. Der Rückgabewert ist einem Variablennamen zuzuweisen, der wieder intern bei der Optimierung weiterverwendet wird. Auch bei der Anlage der Funktionen ist die Reihenfolge, in der sie wegen Abhängigkeiten auszuwerten sind, zu berücksichtigen. Wie bei der Liste der Komponenten gibt es auch hier die Möglichkeit, die Reihenfolge nachträglich durch Pfeil-Buttons zu korrigieren.

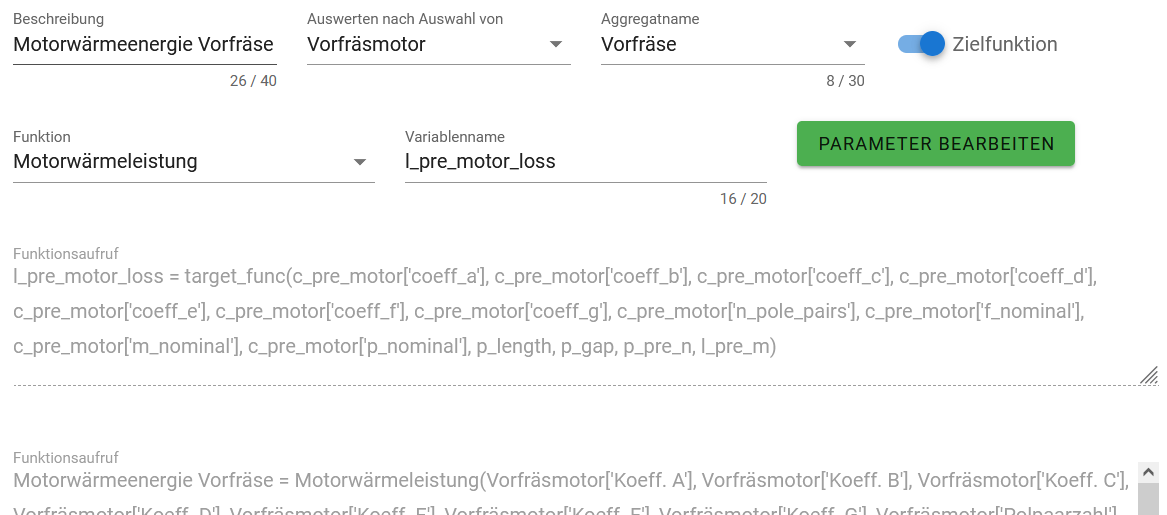


Abbildung 23: Definition von Funktionsaufrufen. Es müssen zu verwendende Funktion und Übergabe-Parameter angegeben werden. Der definierte Aufruf wird unten im Formular in Variablen und unter Angabe der Anzeigenamen dargestellt. Die Eingabe der Aufrufparameter erfolgt über ein untergeordnetes Formular

Die Eingabe der Übergabeparameter für einen Funktionsaufruf erfolgt in einem weiteren Formular (Abbildung 24). An dieser Stelle wird als Referenz auch die zuvor bei der Anlage der Funktion eingegebene Dokumentation angezeigt, damit die Reihenfolge der Parameter leichter eingehalten werden kann. Es kann aus der Liste der im ersten Reiter definierten Prozessparameter und Zwischenergebnissen aus zuvor definierten Funktionsaufrufen sowie aus Komponenteneigenschaften ausgewählt werden. Hierbei stehen alle Eigenschaften von zuvor in der Variantendefinition ausgewählten Komponententypen mit ihren in der Datenbank eingetragenen Spalten zur Verfügung. Es können auch boolsche und numerische Werte als Übergabewerte definiert werden.

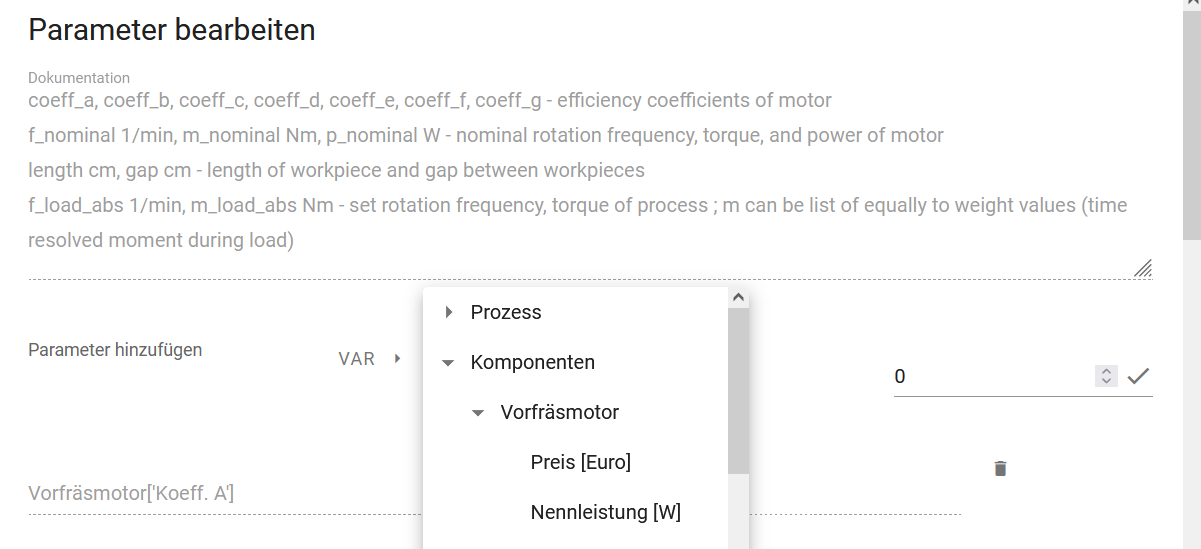


Abbildung 24: Formular zu Eingabe von Parametern. Zuvor angelegte Prozess- und Komponentenparameter können aus einem Drop-Down-Menü als Übergabeparameter ausgewählt werden. Es können auch boolsche oder numerische Werte übergeben werden.

Ebenfalls im Reiter der Variantendefinition werden die Randbedingungen angegeben (Abbildung 25). Auch hier ist zu definieren, nach der Auswahl welcher Komponenten die Bedingung ausgewertet werden kann. Und auch hier ist bei der Anlage eine Reihenfolge zu beachten, die nachträglich korrigiert werden kann. Die Eingabe der Bedingung selbst als Ungleichung erfolgt in einem Formeleditor, in dem auch wieder Prozess- und Komponentenparameter sowie Zwischenergebnisse als Variablen ausgewählt werden können.

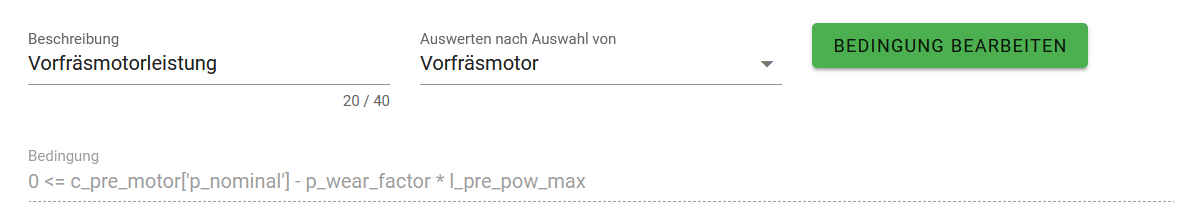


Abbildung 25: Formular zur Anlage von Randbedingungen. Die Eingabe der Bedingung selbst erfolgt durch einen extra zu öffnenden Formeleditor.

Die komplette Anlage von Komponenten und Funktionsaufrufen muss für jede Variante getrennt erfolgen. Um dabei die Arbeit gering zu halten, besteht die Möglichkeit, bereits angelegte Varianten zu kopieren und in diesen Komponenten und Funktionsaufrufe zu entfernen oder weitere hinzuzufügen.

Nach der Definition der Varianten mit ihren Komponenten, Funktionsaufrufen und Bedingungen erfolgt im vierten Reiter die Definition der Variantenauswahl. Wurde nur eine Variante angelegt, wird dieser Schritt übersprungen. Je nachdem, was im ersten Reiter beim Variantenauswahlschema angegeben wurde, erfolgt hier die Eingabe als Definition eines Fragebaums oder einer Frageliste. Für den Baum wurde das Formular bereits in Abbildung 18 dargestellt. Zu jedem angelegten Frageknoten kann angegeben werden, wie die Frage lautet und welche Varianten bei der Beantwortung der Frage mit ja oder nein ausgeschlossen werden. Für die Frageliste erfolgt die Definition entsprechend mit dem Unterschied, dass einzelne Fragen nicht aufeinander aufbauen und bei nachfolgenden Fragen nicht bereits automatisch Optionen ausgeschlossen sind.

Im fünften Reiter kann ein auf das Anwendungsproblem optimierter Lösungsalgorithmus als Python-Code in das entsprechende Formularfeld eingetragen werden. Auch hier muss wieder auf die Formatierung des eingegebenen Codes geachtet werden. Die Eingabe aber optional, wird hier nicht der entsprechende Schieberegler aktiviert, erfolgt die Optimierung mit dem im Programm integrierten Algorithmus.

Auf der letzten Seite der Prozessdefinition können noch für den Anwendungsfall spezifische Hilfetexte eingetragen werden. Diese können dann bei der Eingabe einer konkreten Optimierungsanfrage auf der Seite der Prozessparameter oder weiterer Randbedingungen angezeigt werden. Auch hier ist die Eingabe optional.

## Funktionen für Beispiel-Prozess

### Python-Code für Prozessleistung rotierende Frästools (process\_power\_scraper.py)

import numpy as np  
  
  
def target\_func(diameter, num\_teeth, ang\_kappa, ang\_lambda, cutting\_width, depth\_of\_cut, kc05, rpm, feed\_speed,  
 length\_wp, gap\_wp, r\_bend, n\_motors):  
 *# diameter mm, num\_teeth, ang\_kappa °, ang\_lambda ° - tool parameters, provided as process parameters  
 # cutting\_width mm, depth\_of\_cut mm -  
 # kc05 N/mm^1,5, rpm, feed\_speed m/min, length\_wp cm, gap\_wp cm, r\_bend mm, n\_motors for multiple  
 # returns list of cutting force, moment, max power, and mean power -- accessible by index starting with 0* power\_variables = straight\_cut(diameter, num\_teeth, ang\_kappa, ang\_lambda, cutting\_width, depth\_of\_cut, kc05, rpm,  
 feed\_speed) if r\_bend == 0 else \  
 bend\_cut(diameter, num\_teeth, ang\_lambda, kc05, rpm, feed\_speed, r\_bend)  
 portion\_load = length\_wp / (length\_wp + gap\_wp)  
 return [\*power\_variables, power\_variables[2] \* portion\_load \* n\_motors]  
  
  
def straight\_cut(diameter, num\_teeth, ang\_kappa, ang\_lambda, cutting\_width, depth\_of\_cut, kc05, rpm, feed\_speed):  
 milling\_tool = RotatingTool(diameter=diameter, num\_teeth=num\_teeth, ang\_kappa=ang\_kappa, ang\_lambda=ang\_lambda)  
 milling\_tool.define\_cutting\_process(kc05, rpm, feed\_speed, cutting\_width, depth\_of\_cut)  
 geometry\_variables = milling\_tool.calculate\_geometry\_parameters()  
 return milling\_tool.calculate\_process\_output(geometry\_variables)  
  
  
def bend\_cut(diameter, num\_teeth, ang\_lambda, kc05, rpm, feed\_speed, r\_bend):  
 n = 10  
 xs = [r\_bend \* (1 - 1 / 2 / n - i / n) for i in range(n)]  
 kappa = [np.arccos(x / r\_bend) \* 180 / np.pi for x in xs[::-1]]  
 pieces = [{"ang\_kappa": k, "depth\_of\_cut": (r\_bend \* (1 - np.sin(k \* np.pi / 180)))} for k in kappa]  
 pvs = []  
 for piece in pieces:  
 milling\_tool = RotatingTool(diameter=diameter, num\_teeth=num\_teeth, ang\_kappa=piece["ang\_kappa"],  
 ang\_lambda=ang\_lambda)  
 milling\_tool.define\_cutting\_process(kc05, rpm, feed\_speed, r\_bend / n, piece["depth\_of\_cut"])  
 geometry\_variables = milling\_tool.calculate\_geometry\_parameters()  
 pvs.append(milling\_tool.calculate\_process\_output(geometry\_variables))  
 return [sum([pv[0] for pv in pvs]), sum([pv[1] for pv in pvs]), sum([pv[2] for pv in pvs])]  
  
  
class RotatingTool:  
 *"""Base class for rotating cutting tools."""* def \_\_init\_\_(self, diameter, num\_teeth, ang\_kappa, ang\_lambda):  
 *"""  
 Initialize a rotary cutting tool.  
  
 Parameters  
 ----------  
 diameter : float  
 tool diameter in mm.  
 num\_teeth : integer  
 number of teeth.  
 ang\_kappa : float  
 cutting angle in deg.  
 ang\_lambda : float  
 cutting angle in deg.  
  
 Returns  
 -------  
 Object.  
  
 """* self.d = diameter / 1000  
 self.z = num\_teeth  
 self.ang\_kap = ang\_kappa \* np.pi / 180  
 self.ang\_lam = ang\_lambda  
 self.process\_params = None  
 self.ppp = False  
  
 def define\_cutting\_process(self, kc05, rpm, feed\_speed, cutting\_width,  
 depth\_of\_cut, protrusion=0):  
 *"""  
 Definition of a wood cutting process.  
  
 Parameters  
 ----------  
 kc05: float  
 specific cutting force of material in N/mm^1.5  
 rpm : float  
 rotations per minute.  
 feed\_speed : float  
 feed speed in meters per minute.  
 cutting\_width : float  
 cutting width in mm.  
 depth\_of\_cut : float  
 depth of cut in mm.  
 protrusion : float  
 saw blades only: blade protrusion in mm  
 Returns  
 -------  
 A dictonary of the cutting parameters.  
  
 """  
 # check for numeric values?  
 # conversion to SI Units* parameter\_dict = dict([("kc05", kc05\*1000\*\*1.5), ("rpm", rpm),  
 ("feed\_speed", feed\_speed/60),  
 ("cw", cutting\_width/1000),  
 ("ae", depth\_of\_cut/1000),  
 ("n", rpm/60), ("protrusion", protrusion/1000)])  
  
 self.process\_params = parameter\_dict  
 self.ppp = True  
  
 def check\_process\_param\_dict(self):  
 *"""Check if a Process Parameters Dictonary is available."""* if self.ppp:  
 return self.ppp  
 else:  
 print("Please parse a process parameter dict \  
 to make this calculation.")  
 return self.ppp  
  
 def calculate\_geometry\_parameters(self):  
 *"""  
 Calculate specific geometry values for the given process parameters.  
  
 Returns  
 -------  
 A dictonary containing all relevant cutting process values.  
  
 """* fz = self.fz()  
 vc = self.vc()  
 omega = self.angular\_vel()  
 phi = self.ang\_phi()  
 sb = self.sb(phi)  
 ze = self.ze(phi)  
 hm = self.hm(fz, sb)  
  
 res = dict([("vc", vc), ("feed per tooth", fz), ("phi", phi),  
 ("cut arc length", sb), ("ze", ze),  
 ("medium chip thickness", hm), ("ang\_v", omega)])  
 return res  
  
 def calculate\_process\_output(self, geometry\_dict):  
 *"""  
 Calculate cutting force, moment and power.  
  
 Returns  
 -------  
 A dictonary containing calculation results.  
  
 """* Fc = self.Fc(geometry\_dict.get("medium chip thickness"))  
 M = self.cutting\_moment(Fc, geometry\_dict.get("ze"))  
 P = self.cutting\_power(M, geometry\_dict.get("ang\_v"))  
  
 res = dict([("cutting force", Fc), ("cutting moment", M),  
 ("cutting power", P)])  
   
 return [Fc, M, P]  
  
 def angular\_vel(self):  
 *"""Calculate the tools angular velocity."""* if self.check\_process\_param\_dict():  
 return 2 \* np.pi \* self.process\_params.get("n")  
  
 def vc(self):  
 *"""Calculate the cutting speed."""* if self.check\_process\_param\_dict():  
 return self.d \* np.pi / self.process\_params.get("n")  
  
 def fz(self):  
 *"""Calculate the feed per tooth."""* if self.check\_process\_param\_dict():  
 return self.process\_params.get("feed\_speed") / \  
 (self.process\_params.get("n") \* self.z)  
  
 def sb(self, ang\_phi):  
 *"""Arc length of tooth in cutting operation."""* if self.check\_process\_param\_dict():  
 return self.d \* ang\_phi \* 0.5  
  
 def ze(self, ang\_phi):  
 *"""Calculate the number of active teeth."""* if self.check\_process\_param\_dict():  
 return self.z \* ang\_phi / np.pi \* 0.5  
  
 def ang\_phi(self):  
 *"""  
 Calculate angle of operation?.  
  
 Returns  
 -------  
 Angle of operation in radians.  
  
 """* if self.check\_process\_param\_dict():  
 h1 = self.process\_params.get("ae") - \  
 self.process\_params.get("protrusion")  
 return np.arccos(1-2\*h1/self.d) - \  
 np.arccos(1-2\*self.process\_params.get("protrusion")/self.d)  
  
 def hm(self, feed\_per\_tooth, cut\_arc\_length):  
 *"""  
 Calculate medium chip thickness.  
  
 Parameters  
 ----------  
 feed\_per\_tooth : float  
 feed per tooth in meters/tooth.  
 cut\_arc\_length : floath  
 tooth active arc length in meters.  
  
 Returns  
 -------  
 None.  
  
 """* if self.check\_process\_param\_dict():  
 return feed\_per\_tooth \* np.sin(self.ang\_kap) \* \  
 self.process\_params.get("ae") / cut\_arc\_length  
  
 def Fc(self, medium\_chip\_thickness):  
 *"""Calculate cutting force."""* return self.process\_params.get("kc05") \* \  
 self.process\_params.get("cw") /\  
 np.sin(self.ang\_kap) \* np.sqrt(medium\_chip\_thickness)  
  
 def cutting\_moment(self, cutting\_force, ze):  
 *"""Moment of the cutting force acting on the motor."""* return ze \* cutting\_force \* self.d \* 0.5  
  
 def cutting\_power(self, moment, angular\_vel):  
 *"""Power of the cutting motor."""* return moment \* angular\_vel  
  
  
*# # example usage  
# fuegefraeser = RotatingTool(diameter=125, num\_teeth=3,  
# ang\_kappa=90, ang\_lambda=65)  
  
# fuegefraeser.define\_cutting\_process(kc05=15.5, rpm=9000, feed\_speed=30,  
# cutting\_width=19, depth\_of\_cut=1)  
  
# geometry\_variables = fuegefraeser.calculate\_geometry\_parameters()  
# power\_variables = fuegefraeser.calculate\_process\_output(geometry\_variables)  
  
# print(geometry\_variables, power\_variables)*

### Python-Code für am Motor abfallende Leistung (loss\_motor.py)

from statistics import mean  
from functools import lru\_cache  
  
  
@lru\_cache(200)  
def target\_func(coeff\_a, coeff\_b, coeff\_c, coeff\_d, coeff\_e, coeff\_f, coeff\_g, pole\_pairs, f\_nominal, m\_nominal,  
 p\_nominal, length, gap, f\_load\_abs, m\_load\_abs):  
 *# coeff\_a, coeff\_b, coeff\_c, coeff\_d, coeff\_e, coeff\_f, coeff\_g - efficiency coefficients of motor  
 # pole\_pairs, f\_nominal 1/min, m\_nominal Nm, p\_nominal W - number of pole pairs, nominal frequency, torque, and power of motor  
 # length cm, gap cm - length of workpiece and gap between workpieces  
 # f\_load\_abs 1/min, m\_load\_abs Nm - set rotation frequency, torque of process ; m can be list of equally to weight values (time resolved moment during load)* f\_load = f\_load\_abs / f\_nominal \* pole\_pairs / 60  
 f\_idle = f\_load  
 m\_load = [load / m\_nominal for load in m\_load\_abs] if type(m\_load\_abs) == list else m\_load\_abs / m\_nominal  
 portion\_load = length / (length + gap)  
 loss\_load = mean(  
 [motor\_loss(coeff\_a, coeff\_b, coeff\_c, coeff\_d, coeff\_e, coeff\_f, coeff\_g, f\_load, m) for m in m\_load]) if type(  
 m\_load) == list else \  
 motor\_loss(coeff\_a, coeff\_b, coeff\_c, coeff\_d, coeff\_e, coeff\_f, coeff\_g, f\_load, m\_load)  
 relative\_loss = portion\_load \* loss\_load + (1 - portion\_load) \* motor\_loss(coeff\_a, coeff\_b, coeff\_c, coeff\_d,  
 coeff\_e, coeff\_f, coeff\_g, f\_idle, 0)  
 return relative\_loss \* p\_nominal  
  
  
def motor\_loss(coeff\_a, coeff\_b, coeff\_c, coeff\_d, coeff\_e, coeff\_f, coeff\_g, f, m):  
 return coeff\_a + coeff\_b \* f + coeff\_c \* f \*\* 2 + coeff\_d \* f \* m \*\* 2 + coeff\_e \* f \*\* 2 \* m \*\* 2 + coeff\_f \* m \*\* 2 + coeff\_g \* m  
  
  
def motor\_loss\_eight(p\_100\_100, p\_50\_100, p\_0\_100, p\_100\_50, p\_50\_50, p\_0\_50, p\_50\_25, p\_0\_25, f\_rel, m\_rel):  
 f\_z = f\_rel \* 100 *# in %* m\_z = m\_rel \* 100 *# in %* if f\_z < 50 and m\_z < 50: *# seg 3* return p\_0\_25 + f\_z / 50 \* (p\_50\_25 - p\_0\_25) + (m\_z - 25) / 25 \* (  
 p\_0\_50 - p\_0\_25 + f\_z / 50 \* (p\_50\_50 - p\_0\_50 - p\_50\_25 + p\_0\_25))  
 if f\_z < 50 and m\_z >= 50: *# seg 1* return p\_0\_50 + f\_z / 50 \* (p\_50\_50 - p\_0\_50) + (m\_z - 50) / 50 \* (  
 p\_0\_100 - p\_0\_50 + f\_z / 50 \* (p\_50\_100 - p\_0\_100 - p\_50\_50 + p\_0\_50))  
 if f\_z >= 50 and m\_z < 50: *# seg 4* return p\_0\_25 + f\_z / 50 \* (p\_50\_25 - p\_0\_25) + (m\_z - 25) / 25 \* (  
 p\_50\_50 - p\_0\_25 + (f\_z - 50) / 50 \* (p\_100\_50 - p\_50\_50) - f\_z / 50 \* (p\_50\_25 - p\_0\_25))  
 if f\_z >= 50 and m\_z >= 50: *# seg 2* return p\_50\_50 + (f\_z - 50) \* f\_z / 50 \* (p\_100\_50 - p\_50\_50) + (m\_z - 50) / 50 \* (  
 p\_50\_100 - p\_50\_50 + (f\_z - 50) / 50 \* (p\_100\_100 - p\_50\_100 - p\_100\_50 + p\_50\_50))

### Python-Code für die am Spannungswandler abfallende Wärmeleistung (loss\_converter.py)

from statistics import mean  
  
  
def target\_func(p\_90\_100, p\_50\_100, p\_0\_100, p\_90\_50, p\_50\_50, p\_0\_50, p\_50\_25, p\_0\_25, p\_nominal, f\_nominal, length,  
 gap, f\_load\_abs, m\_load\_abs, m\_nominal\_motor):  
 *# p\_x\_y - relative efficiency values of motor  
 # p\_nominal W, f\_nominal Hz - nominal power and rotation frequency of motor  
 # length cm, gap cm - length of workpiece and gap between workpieces  
 # f\_load\_abs 1/min, m\_load\_abs Nm, m\_nominal Nm - set rotation frequency, torque of process, and nominal torque of motor  
 # m\_load\_abs can be list of equally to weight values (time resolved moment during load)* f\_load = f\_load\_abs / f\_nominal / 60  
 f\_idle = f\_load  
 i\_qz\_load = [load / m\_nominal\_motor for load in m\_load\_abs] if type(  
 m\_load\_abs) == list else m\_load\_abs / m\_nominal\_motor  
 portion\_load = length / (length + gap)  
 loss\_load = mean(  
 [converter\_loss(p\_90\_100, p\_50\_100, p\_0\_100, p\_90\_50, p\_50\_50, p\_0\_50, p\_50\_25, p\_0\_25, f\_load, i\_qz\_load) for m  
 in i\_qz\_load]) if type(i\_qz\_load) == list else \  
 converter\_loss(p\_90\_100, p\_50\_100, p\_0\_100, p\_90\_50, p\_50\_50, p\_0\_50, p\_50\_25, p\_0\_25, f\_load, i\_qz\_load)  
 relative\_loss = portion\_load \* loss\_load + (1 - portion\_load) \* converter\_loss(p\_90\_100, p\_50\_100, p\_0\_100, p\_90\_50,  
 p\_50\_50, p\_0\_50, p\_50\_25, p\_0\_25,  
 f\_idle, 0)  
 return relative\_loss \* p\_nominal / 100  
  
  
def converter\_loss(p\_90\_100, p\_50\_100, p\_0\_100, p\_90\_50, p\_50\_50, p\_0\_50, p\_50\_25, p\_0\_25, f\_rel, i\_rel):  
 f\_z = f\_rel \* 100 *# in %* i\_qz = i\_rel \* 100 *# in %* if f\_z < 50 and i\_qz < 50: *# seg 3* return p\_0\_25 + f\_z / 50 \* (p\_50\_25 - p\_0\_25) + (i\_qz - 25) / 25 \* (  
 p\_0\_50 - p\_0\_25 + f\_z / 50 \* (p\_50\_50 - p\_0\_50 - p\_50\_25 + p\_0\_25))  
 if f\_z < 50 and i\_qz >= 50: *# seg 1* return p\_0\_50 + f\_z / 50 \* (p\_50\_50 - p\_0\_50) + (i\_qz - 50) / 50 \* (  
 p\_0\_100 - p\_0\_50 + f\_z / 50 \* (p\_50\_100 - p\_0\_100 - p\_50\_50 + p\_0\_50))  
 if f\_z >= 5 and i\_qz < 50: *# seg 4* return p\_0\_25 + f\_z / 50 \* (p\_50\_25 - p\_0\_25) + (i\_qz - 25) / 25 \* (  
 p\_50\_50 - p\_0\_25 + (f\_z - 50) / 40 \* (p\_90\_50 - p\_50\_50) - f\_z / 50 \* (p\_50\_25 - p\_0\_25))  
 if f\_z >= 50 and i\_qz >= 50: *# seg 2* return p\_50\_50 + (f\_z - 50) \* f\_z / 40 \* (p\_90\_50 - p\_50\_50) + (i\_qz - 50) / 50 \* (  
 p\_50\_100 - p\_50\_50 + (f\_z - 50) / 40 \* (p\_90\_100 - p\_50\_100 - p\_90\_50 + p\_50\_50))

### Python-Code für Getriebe-Leistungen und –momente (gears.py)

from math import pi  
  
  
*# i\_gears Uebersetzung Getriebe, eta Getriebewirkungsgrad, m\_nominal Nennmoment Getriebe Nm,  
# disc\_diam m, v\_feed Vorschub m/min, process\_pow  
# returns gear\_loss, total moment of feed system, input rotation rate 1/min and process moment*def target\_func(i\_gears, eta, m\_nominal, disc\_diam, v\_feed, process\_pow):  
 v\_feed\_si = v\_feed / 60  
 omega = dg(i\_gears, v\_feed\_si, disc\_diam)  
 m\_process = process\_moment(process\_pow, omega)  
 m\_gr = gear\_moment(m\_nominal, m\_process, eta)  
 gl = gear\_loss(m\_gr, omega)  
 total\_moment = moment\_feed\_system(m\_gr, m\_process)  
 return [gl, total\_moment, omega \* 30 / pi, m\_process]  
  
  
*# Getriebeverlustleistung*def gear\_loss(m\_gr, omega):  
 return m\_gr \* omega  
  
  
*# m\_g Getriebenennmoment aus DB*def moment\_feed\_system(m\_g, m\_process):   
 return m\_g + m\_process  
  
  
*# m\_nominal Nennmoment Getriebe*def gear\_moment(m\_nominal, m\_process, eta): *# oben berechnete Prozessleistung muss für m\_process durch die Ausgangsdrehzahl des Getriebes geteilt werden, Drehzahl aus Vorschubgeschwindkeit und Durchmesser der Scheibe (ebenfalls als Prozessparameter anzugeben)* return (1 - eta) / eta \* (0.6 \* m\_nominal + 0.4 \* m\_process)  
  
  
*# proc\_pow aus process\_power()*def process\_moment(process\_pow, omega):  
 return process\_pow / omega  
  
  
*# Antriebsdrehzahl / Getriebeeingangsdrehzahl (omega)  
# i Übersetzung aus Datenbank  
# f Drehzahl aus f\_abtrieb()*def dg(i\_gears, v\_feed, disc\_diam):  
 f = omega\_abtrieb(v\_feed, disc\_diam)  
 return i\_gears \* f  
  
  
*# Getriebeabtriebsdrehzahl*def omega\_abtrieb(v\_feed, d):  
 return 2 \* v\_feed / d

### Python-Code für Reibungsverluste am Vortrieb (feed.py)

from math import sqrt, cos, sin, atan  
  
*# a Abstand Anpressrollen, mu Reibungskoeff. Band, k\_c Federkonstante, f\_zero Vorspannung Feder,  
# v\_feed Vorschubgeschw. m/min, x\_1, y\_1, dx, x\_3, y\_3, length\_band, length\_wp, gap\_wp  
# alle Eingabewerte in SI-Einheiten  
# Verlustleistung Motor --- fuer Leerlauf Getriebereibverluste einsetzen  
# Koeffizienzen Motor, Gesamtmoment (relativ), relative Drehzahl*def target\_func(a, mu, k\_c, f\_zero, v\_feed, x\_1, y\_1, dx, x\_3, y\_3, length\_band, length\_wp, gap\_wp):  
 v\_feed\_si = v\_feed / 60  
 return process\_power(a, mu, k\_c, f\_zero, v\_feed\_si, x\_1, y\_1, dx, x\_3, y\_3, length\_band, length\_wp, gap\_wp)  
  
  
def process\_power(a, mu, k\_c, f\_zero, v\_feed, x\_1, y\_1, dx, x\_3, y\_3, length\_band, length\_wp, gap\_wp):   
 s = spring\_path(x\_1, y\_1, dx, x\_3, y\_3)  
 l = length\_band \* length\_wp / (length\_wp + gap\_wp) *# l: summierte Laenge aller Werkstuecke im System* return (mu \* s \* k\_c \* l / a + f\_zero) \* v\_feed  
   
  
def spring\_path(x\_1, y\_1, dx, x\_3, y\_3): *# Federweg s* alpha = angle(x\_1, y\_1, dx)  
 return (sqrt(x\_3 \*\* 2 + y\_3 \*\* 2) \* cos(alpha) \*\* 3 \* (x\_3 - sin(alpha) \* sqrt(x\_3 \*\* 2 + y\_3 \*\* 2))) /\  
 sqrt(x\_1 \*\* 2 + y\_1 \*\* 2 - (x\_1 - dx) \*\* 2)  
  
  
def angle(x\_1, y\_1, dx):  
 return atan((x\_1 - dx) / sqrt(x\_1 \*\* 2 + y\_1 \*\* 2 - (x\_1 - dx) \*\* 2))

### Python-Code für Zwischengrößen für Kappaggregat (trim\_values.py)

from math import tan, radians  
  
  
def target\_func(v\_feed, alpha, length, gap, width\_band):  
 *# v\_feed m/min, alpha °, length cm, gap cm, width\_band mm  
 # returns downward speed in m/min, load and idle time in arbitrary units* v\_down = tan(radians(alpha)) \* v\_feed  
 t\_total = (length + gap) / v\_feed  
 t\_load = width\_band / v\_down  
 t\_idle = t\_total - t\_load  
 return [v\_down, t\_load, t\_idle]

### Python-Code für Prozessleistung an Abziehklingen (process\_power\_scraper.py)

from math import sin, sqrt  
  
  
def target\_func(ang\_kappa, kc05, feed\_speed, cutting\_width, depth\_of\_cut, length, gap):  
 *# ang\_kappa °, kc05 N/mm^1.5, feed\_speed m /min, cutting\_width mm, depth\_of\_cut mm,  
 # length workpiece cm, gap workpiece cm* power\_load = feed\_speed/60 \* kc05\*1000\*\*1.5 \* cutting\_width/1000 \* sqrt(depth\_of\_cut/1000) / sin(ang\_kappa)  
 return length / (length + gap) \* power\_load

### Python-Code für Wäremeleistung für Gleichgewichtszustand Kleberbehälter (glue.py)

def target\_func(lamb, a, d):  
 return lamb \* a \* 180 / d / 0.993  
  
*# lambda: specific thermal conductivity W/m/K  
# a, d: surface, thickness glue compartment*

### Python-Code für Kantenbandbreite am Bündigfräsaggregat (overlap.py)

def target\_func(width\_wp, width\_band):  
 *# width workpiece mm, width edge band mm* return (width\_band - width\_wp) / 2

### Python-Code Hilfsfunktion zur Parameterauswahl (select\_from\_list.py)

def target\_func(list\_var, index):  
 *# list, index* return list\_var[index]