Integration of Geometric Tolerance Analysis in System Simulations via Functional Mock-up Units

Christoph Steinmann[1] Konstantin Wrede[2] Jens Schirmer[1] Jens Lienig[1]

[1]Institute of Electromechanical and Electronic Design, Dresden University of Technology, Germany
christoph.steinmann@tu-dresden.de
[2] Fraunhofer Institute for Integrated Circuits IIS, Division Engineering of Adaptive Systems EAS, Dresden, Germany
konstantin.wrede@eas.iis.fraunhofer.de

Abstract

This paper presents a concept for integrating geometric tolerance analysis into system simulations using Functional Mock-up Units (FMUs). While various methods and tools for tolerance analysis exist in the mechanical domain, there is currently no standardized or widely established approach for their integration into multi-domain system simulation. This work proposes a structured FMU interface based on the Functional Mock-up Interface (FMI) standard, enabling a modular and reusable representation of tolerance behaviour. The concept is demonstrated in a case study, in which an FMU for static tolerance calculation was implemented and successfully verified against a commercial analysis tool. Furthermore, the use of the FMU with FMUGym illustrates the potential of the FMI ecosystem to flexibly combine tolerance models with other simulation environments and analysis methods. The results demonstrate that FMUs can provide a suitable and tool-independent interface for integrating geometrical tolerance effects into system-level simulations and model-based engineering processes.

Keywords: geometric tolerances, ISO GPS, tolerance analysis, MBSE, system simulation, Functional Mock-up Interface

1 Introduction

Geometrical tolerances of mechanical parts, in particular shape and position tolerances, play a central role in the precise definition of technical systems. They are defined by the geometric product specifications (GPS). Although there are already numerous simulation approaches for these tolerances in the mechanical domain, there is a lack of standardised approaches for integrating them into multi-domain simulations which are used for Model Based System Engineering (MBSE). The aim of this work is therefore to investigate a possible solution to enable the integration of geometrical tolerances for such simulations and to evaluate Functional Mock-up Units (FMU) (Blochwitz et al. 2011) as a possible interface. FMUs could serve as the connection between commercial Computer-Aided Tolerance (CAT) management programs or other analysis models and the system simulation, as shown conceptually in Figure 1.

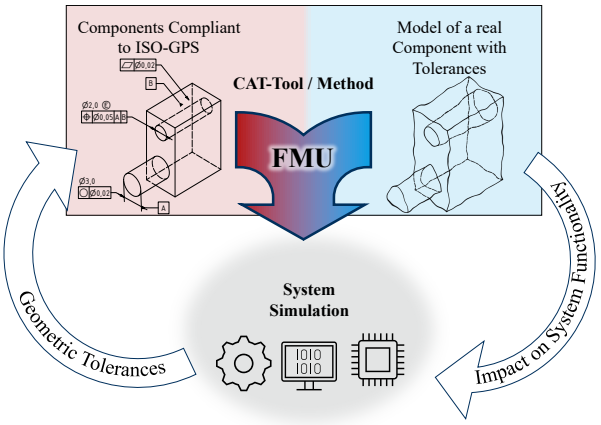

Figure 1. Our concept for the integration of geometric tolerance simulations into the product development process with system simulation via FMUs, influenced by ISO-GPS (red) and the modelling method used (blue).

Several challenges show the necessity of such an approach. Firstly, CAT with commercial tools is often regarded as a black box by the user, as the underlying methods are not accessible in detail (Kosec, Skec, and Miler 2020). This makes it difficult to adapt the models to new applications under changed boundary conditions. In addition, there is still no standardised methodology for setting up mechanical tolerance simulations, which impairs the comparability and reproducibility of results (Morse et al. 2018). The usability of existing scientific solutions also poses a problem, as many approaches are either too complex, not accurate enough for certain problems or not sufficiently user-friendly (Walter et al. 2021). There are also approaches from the field of MBSE to solve geometric problems using library elements (Moers et al. 2025). However, these are much less flexible and less precise than tolerance simulations that are specifically tailored to the problem of mechanical part geometry.

Precise tolerance simulation requires expert knowledge and a deep understanding of the respective domain (Ramnath et al. 2018). Modularisation of the modelling task therefore appears to be a promising approach to reduce complexity and facilitate access for users.

This work focuses on the simulation of rigid mechanical assemblies, with the aim of enabling static tolerance

analysis in the context of system simulation - similar to existing commercial tools such as *CETOL*, *MECAmaster*, or *3DCS*. While dynamic and transient simulations of tolerances have been addressed in initial research (Husch and Walter 2024), this paper deliberately focuses on static scenarios as a first step. Future extensions could build on these results to support more complex, dynamic applications.

The main contributions of this paper are:

- An analysis of the state of the art in tolerance simulation methods, with a focus on their interfaces and applicability within system simulation.
- The introduction of a new FMU-based concept for transferring specialised tolerance analysis methods to system-level simulations, demonstrated through a case study.
- The demonstration of additional use cases for FMUs within the FMI ecosystem, illustrated by performing a Monte Carlo simulation using the FMUGym toolbox.

2 Related Work and Methods

In order to effectively implement a mechanical tolerance model, it is essential to first consider both the standards and regulations that define geometric specifications, as well as the scientific methods used to simulate and analyse them. The relevant aspects of FMUs in relation to this application are important as well and thus, are also summarized below.

2.1 Mechanical Tolerance Analysis

Mechanical tolerance simulation in this paper's context specifically refers to the simulation of geometric deviations in components arising from their dimensions (size), shape (form), orientation, position (location), profile, and run-out in relation to one another. The underlying standard is the ISO-GPS system (ISO International Organization for Standardization 2011). The unique aspect here is that these tolerances go beyond pure scalar dimensions and therefore require special calculation methods for linking several tolerances.

Before the established analysis models are presented, two terms that are equally relevant for all of the methods need to be explained:

- *Functional Requirement (FR):* This is sometimes also referred to as a functional condition and describes the geometric entity that is relevant for the correct function of the assembly in the technical application and has to stay inside of certain limits. Depending on the calculation method used, this describes either a scalar (a distance) or the three-dimensional boundary of a resulting position in space. Defining this condition is the task of the user for all available methods. There are approaches for systematically or even automatically determining this condition, which can be used when solving specific tasks (Anselmetti 2006). FRs still have to be interpreted afterwards. Frequent target variables are, for example, the clearance of a fit or the misalignment in an assembly. The FR is an edge in the contact graph.
- *Contact Graph:* Or assembly graph is used to model the network of mating contacts and constraints between features in an assembly. This graph defines the relationships that form the tolerance chains - the paths through which individual part variations accumulate to affect a functional requirement. In commercial CAT software, the contact graph can be derived from the 3D CAD model, sometimes with the help of additional user input. For manual analyses according to the methods presented below, it must be created by the user, for which systematics are suggested in the literature (Haghighi et al. 2014).

The following paragraphs describe six analysis methods as well as groupings of tolerance models with their primary literature sources. There exist various modifications of most of the methods, for which numerous reviews (Hallmann, Schleich, and Wartzack 2020; Chen et al. 2014; Cao, Liu, and Yang 2018; Shah et al. 2007; Shen et al. 2005) provide a good overview. A summary of the most relevant aspects can be found in Table 1.

Vector Loop
(Chase et al. 1996)
The Vector Loop method models assemblies as closed kinematic loops, using vectors to represent nominal dimensions and constraints for mating conditions between parts. Tolerances are represented by small adjustments at the joints. Nominal dimensions, tolerance values, and assembly constraints are required as inputs. The calculated result is a point to point distance, which is interpreted as the FR. The method is able to calculate worst-case as well as statistical results with consideration of input-distributions. It can handle serial tolerance chains well but is not capable to represent complex, parallel tolerance chains in its basic form due to its limited ability to model intricate geometric interactions.

Matrix TTRS
(Desrochers and Riviere 1997)
The Matrix TTRS (Technologically and Topologically Related Surfaces) method uses homogeneous transformation matrices to model tolerances as small permissible displacements. It requires the nominal geometry, tolerance zone constraints, and assembly connections. The output is a range of variation in a critical assembly dimension based on solving inequality constraints. This represents the worst case calculation. It can theoretically handle complex parallel tolerance chains, but solving large constraint sets is computationally expensive, making it impractical for very detailed assemblies.

DOI
10.3384/ecp218765

Table 1. Overview of tolerance analysis models with their relevant characteristics.

Method / Model	*Result*	*Tolerance Representation and Calculation*	*Comment*
Vector Loop	distance, point to point	solution for joint conditions, concatenation of linearized transformation matrices	point based method (not well suited to represent form deviations), no parallel chain calculations
Matrix TTRS	distance, point to point	TTRS with non-linear constraint inequalitys	point based method, statistics only from Monte Carlo
Jacobian Torsor	interval of FRs position and rotation (6D)	multiplication of jacobian matrix with SDT (interval arithemtic)	partly represents form tolerances, handles certain parallel chains
Deviation-Domain	boundaries of tolerance zone in deviation space (6D)	coordinate transformation and polytope calculation	good compatibility with ISO-GPS, complicated implementation
Parametric Variation & Skin Model Shape	one or multiple measurements with statistic distribution	variation of parameters and 3D-contact solving with solids or mesh bodies	requires a sophisticated solution to solve the 3D contact problems

Jacobian Torsor
(Desrochers, Ghie, and Laperrie‘re 2003)
The Jacobian Torsor method combines Jacobian matrix transformations with Small Displacement Torsors (SDTs) to represent tolerances in six degrees of freedom. It requires assembly constraints, SDT representations of tolerances, and a functional requirement definition. The output is the variation range of a key dimension, computed using interval algebra. It handles complex and to some extent also parallel tolerance chains, making it usable for larger assemblies while maintaining computational efficiency.

Deviation Domain Methods
(T-Map (Davidson, Mujezinovic, and Shah 2002) and Polytope (Homri, Teissandier, and Ballu 2015))
The methods working in the Deviation Domain explicitly compute the geometric region of variation using operations on the tolerance space, which represents the possible translation and rotation of a feature (6D). The required input includes tolerance zones, assembly constraints, and mating feature definitions. The output is a geometric volume of possible variations, enabling worst-case and statistical analysis. It can theoretically handle parallel tolerance chains but suffers from exponential computational complexity, making it suitable for small critical assemblies rather than large systems.

Parametric Variation
(Gupta and Turner 1993)
The Parametric Variation method applies Monte Carlo simulation to a parametric CAD or analytical model, varying dimensions based on statistical distributions. It requires a parametrised assembly model, tolerance distributions, and assembly constraints. The output is a probabilistic distribution of functional variation. It can handle complex and parallel tolerance chains within the limits of the used geometric contact solver (for parametric CAD) but may require thousands of runs, making it computationally expensive.

Skin Model Shape
(Schleich et al. 2016)
The Skin Model Shape method (SMS) simulates tolerance effects by generating randomized part surfaces, assembling them virtually, and measuring the outcome. It requires a detailed CAD model, tolerance distributions, and assembly constraints. The output is a statistical distribution of assembly variations, considering realistic form deviations. It can handle complex and parallel tolerance chains, but the computational burden is extremely high, making it impractical for routine design work but useful for high-precision simulations. It can be considered a further development of the parametric variation approach.

All analysis methods have in common that they take information from both the nominal geometry (3D model) and the defined tolerances as input - including positions, interactions according to the contact graph, tolerance types and tolerance zone widths. The output is an FR in one or more dimensions, either as a scalar value or as a value range.

2.2 FMU

An FMU is a self-contained executable file that facilitates integration and simulation within various development environments that adhere to the FMI standard (Blochwitz et al. 2011), such as Modelica based tools or MATLAB Simulink. The idea behind the FMI standard is to enable comprehensive digital modelling of complex systems. If a real-world product consists of multiple interacting com-

ponents, each governed by intricate physical laws and associated control systems - including electronic, hydraulic, and software-based controls - then it should likewise be possible to digitally assemble a virtual counterpart from individual models representing these subsystems. Each FMU is packaged as a zip archive comprising an XML file with static meta-information, and a C library used for evaluating model equations and their derivatives. This library can be provided in source form or as a dynamically linked compiled library (DLL), and is suitable either for static or dynamic linking. FMI defines two primary forms of interconnection: model exchange, in which interactions occur at the level of a differential-algebraic system of equations (DAE), and co-simulation, which handles discrete input-output interactions at specific communication time points. The FMI standard 3.0 further introduces the scheduled execution format supporting concurrent computation of model partitions on a single computational resource (e.g. specific CPU-core). Although primarily designed for transient simulation scenarios, the FMI standard has been applied in previous research to perform steady-state (design-oriented) simulations, typically involving static single-step computations (Gohl et al. 2024). Similar to our intended use-case, FMUs have also been used as transfer interfaces for three dimensional FEM models (Gödecke et al. 2012).

While the FMI standard and FMUs are well suited for modular modelling and simulation, integrating uncertainty handling and automation features remains an active area of development. The Modelica Credibility Library (Otter et al. 2022), for instance, offers structured mechanisms to represent uncertainty directly within Modelica models and supports traceability, credibility assessment, and transparent documentation of assumptions. However, it has not yet been used in our concept for two main reasons: first, the library lacks interfaces for automated simulation workflows or integration with Python-based environments; second, the current data is derived solely from predefined GPS, so detailed traceability is not yet a priority. Recent contributions such as FMUGym (Wrede et al. 2024) have extended FMI applicability to machine learning scenarios. FMUGym is a Python-based interface that supports reinforcement learning-based control of FMUs and accommodates model uncertainty. It handles co-simulation FMUs compatible with FMI 1.0, 2.0, and 3.0, linking simulation environments with reinforcement learning libraries. In a simplified manner, FMUGym also allows a standardized evaluation of simulation model outputs under specified input distributions and varying system parameters. This makes it suitable for analysing the tolerances of technical systems.

3 Proposed Concept for Geometric Tolerance FMUs

This chapter presents how FMUs can be set up for the simulation of geometric tolerances of mechanical components

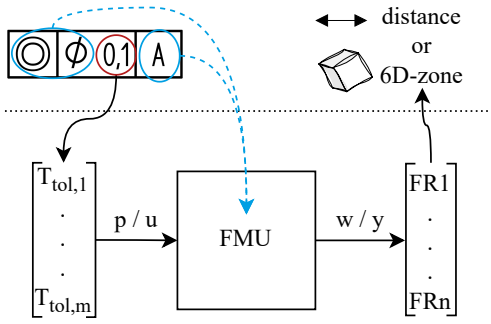

Figure 2. FMU Concept: Tolerance zone size T_{tol} from ISO feature control frame as input and Functional Requirement FR as output depending on the method used for tolerance calculation.

and which interfaces they provide. The actual computation of the tolerance values is not subject of this paper, as this depends heavily on the selected method and the considered application - a brief explanation and assessment of the suitability of various methods can be found in subsection 2.1. The goal is rather to formulate a generic specification of interfaces in line with the FMI standard, in particular with regard to possible inputs and outputs as shown in Figure 2. For the static case considered in this paper, both parameters (p) and input variables (u), as well as local variables (w) and output variables (y), can be treated equivalently to represent mechanical tolerances. It should be noted, that a differentiation between these categories would be necessary in the context of dynamic simulations (e.g. with external forces).

The idea for structuring the interfaces is based on the elements of the tolerance feature control frame according to ISO-GPS. The scalar values of the tolerance field variables are provided as input variables - i.e. the numerical value in the second box of the ISO tolerance field (see Figure 2). Further information, such as the specific tolerance type, geometric datum references or other model-dependent details, are only included in the internal structure of the FMU. This separation makes it possible to keep the external interface stable and generic, while the internal modelling can be flexibly adapted to different analysis methods. The FMU must also be capable of modelling several FRs at the same time, as complex systems can rarely be reduced to a single FR. This results in the need to integrate several analysis models within an FMU. To the outside world, each tolerance defined in the CAD-model or drawing only appears as a single parameter T_{tol} representing the size of the tolerance field. An exemplary realisation of this structure can be found in the case study in section 4.

The output of the FMU are the individual FRs, which can take different forms depending on the model type and analysis objective. In the simplest case, the output is a scalar, e.g. to describe a distance between two points. However, even in these simple approaches, the result is typically given as a range with statistical or worst-case limits for this distance - effectively requiring two output variables per FR to represent minimum and maximum

DOI
10.3384/ecp218765

values. For more complex models such as the Jacobian torsor or the deviation domain, deviation ranges or three-dimensional spaces can be expected as outputs. The actual reduction to interpretable individual values, such as worst-case values, then proceeds outside the FMU. Probability distributions could also theoretically be the output. However, we suggest to carry out the actual statistical evaluation outside of the FMU, as the FMU should only provide the geometric tolerance calculation.

The choice of FMUs as the basis for this approach is suitable in particular because they allow modular, standardised integration into existing simulation environments and promote the reusability of complex analysis logic.

4 Case Study

The following case study shows an example of how a geometric tolerance simulation can be integrated into an FMU and subsequently used as such. The starting point is the analysis of an assembly with the commercial simulation tool CETOL. Based on this analysis, an FMU is created that calculates the same FRs as the reference analysis internally and uses the interfaces described in section 3.

4.1 Application Example

The example shown in Figure 3 shows a simplified model of a stepper motor with translational output. A magnetic field is generated in the stator (S), which causes the magnetic rotor (R) to rotate. The rotor is held between two balls (B) and drives the nut (N) via a spindle (not shown here), which converts the rotary movement into a linear movement. The assembly is completed by the end cap (C), which also serves as a linear bearing for the nut.

The contact graph in Figure 4 illustrates the relationships between the individual components and contains the labels of all relevant connection points, whose interactions are explained in more detail in the following subchapter. The values and types of tolerances are listed in Table 2.

The study focuses on the gap widths $FR1$ and $FR2$ marked in Figure 3. $FR1$ describes the air gap between the stator and the rotor through which the magnetic field flows, while $FR2$ relates to the mechanical fit of the linear bearing on the output. The upstream analysis in CETOL also serves to identify those tolerances within the defined mechanical chain that have a relevant influence on the FR under evaluation. Parameters that are not categorised as relevant in this analysis are not taken into account in the subsequent FMU implementation. This approach is useful as only the relevant parameters should be included, even in the event of a future automated FMU export. For the present case study, this reduces the manual calculation effort when implementing the FMU.

The following additional boundary conditions for the implementation of the FMU result from the example application and the setup of the CETOL analysis. These are particularly relevant in regard to the comparison of the calculations implemented here with the previously presented analysis methods from the literature:

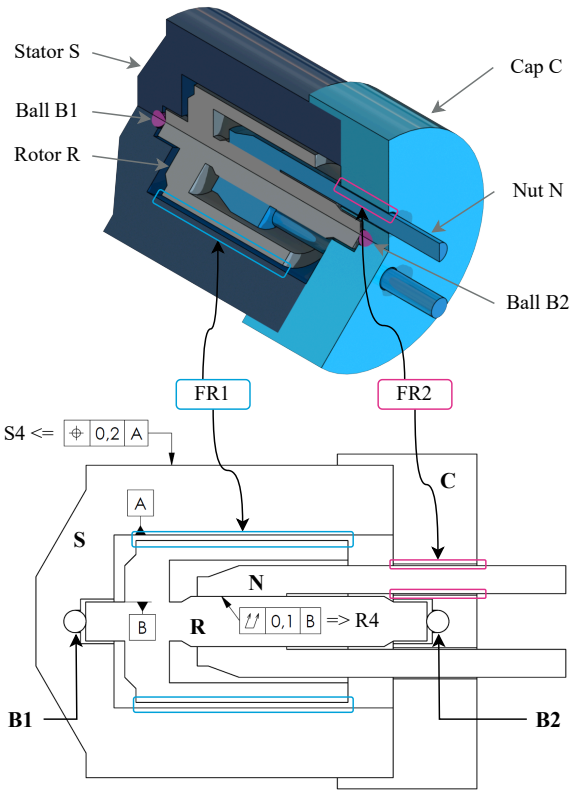

Figure 3. CAD model of a stepper motor with component labels and FR positions used for the case study. The assignment of the labels for the geometric tolerances is shown as an example for S4 and R4.

- Rigid solids without deformation are assumed.
- Contacts between components only occur at defined connection points.
- Shape tolerances are not fully mapped in accordance with ISO standards.
- The assembly sequence specifies manually defined kinematic dependencies between the components.
- Parallel tolerances are simplified by making assumptions about their degrees of freedom.

4.2 Implementation as FMU

In this example, the analysis of the mechanical tolerances is limited to a 2D plane. This simplification is acceptable due to the predominantly rotationally symmetrical components; minor deviations in the non-rotationally symmetrical nut (N) and end cap (C) are accepted for the benefit of clarity. It is not the scope of this paper to demonstrate how complex 3D analysis methods can be automatically translated into executable code, but rather to show the basic process of using an FMU to transfer the tolerance analysis model. Therefore, the internal content of the FMU must still be manually created and verified. A simplified case such as the one presented here is particularly beneficial

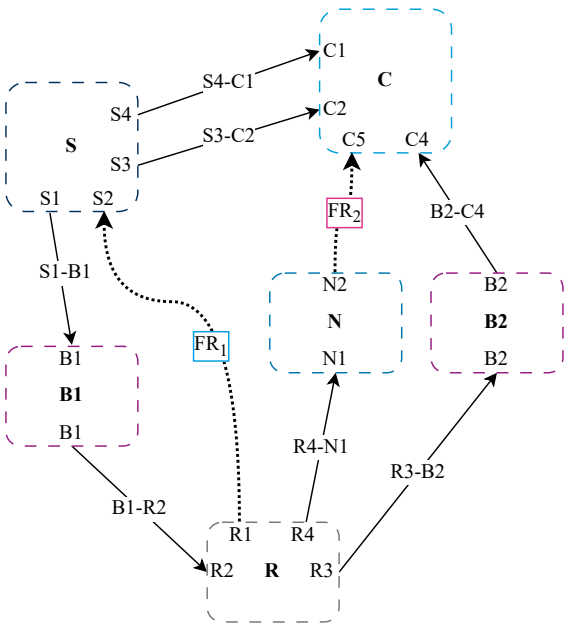

Figure 4. Contact graph of the motor with part labels and links between tolerances

in this context, as it allows for transparent modelling and easier validation of the implemented logic.

As described in subsection 2.1, the existing analysis methods differ fundamentally in their calculation logic. Accordingly, the internal code within an FMU also varies depending on the underlying method. For this case study, the following procedure was chosen to build the FMU:

1. Analysation of the tolerances and joints defined in CETOL
2. Derivation of the kinematic chain and constraints
3. Analytical calculation of the influence of all part tolerances on the investigated FR
4. Implementation of the resulting equations as an Modelica algorithm, including the summation of the final results
5. Export of the Modelica model as FMU

Table 2. Type and size of tolerance zone T_{tol} of all tolerances relevant for calculating the FRs

Tol	*Type*	T_{tol}	*Kinematics*	*FR*
R1	diameter	0.1 mm	linear	$FR1$
S2	diameter	0.1 mm	linear	$FR1$
S4	position	0.2 mm	tilt	$FR1$, $FR2$
C4	profile	0.1 mm	tilt	$FR1$, $FR2$
C5	diameter	0.2 mm	linear	$FR2$
R4	runout	0.1 mm	tilt	$FR2$
N2	diameter	0.1 mm	linear	$FR2$

In combination with the previously defined boundary conditions, the approach is similar to the vector loop method. To simplify the implementation, a solver algorithm for determining the linking states is not used. Instead, all relevant cases are solved analytically and are integrated into the FMU as equations.

For $FR1$ and $FR2$, this results in a compact set of equations for the relevant tolerances listed in Table 2. R1 and S2 describe the simple linear variation of the air gap $FR1$ on both sides. The same applies to C5 and N2 with regard to the fit of $FR2$. The tolerance variations of C4 and S4 result in different tilting movements of the kinematic chain, which influences both FRs. R4 is also a tilting move only affecting $FR2$.

To determine the resulting worst-case tolerance, the effects on the target dimension are calculated separately for each individual tolerance along the kinematic chain - as shown below. Finally, all partial solutions FRn_i are summed up and added to the nominal value of FRn.

Our calculation is based on the radius (y-axis in Figure 5), which is why all diameters are halved. In the linear cases (see Table 2), the tolerances in this example - due to the symmetrical tolerance zones - are each taken into account in the calculation with half the value of their tolerance zone width. The individual contributions thus result directly from the width of their corresponding tolerance zone T_{tol}:

$$FRn_{i,max} = +T_{tol}/4 \tag{1}$$

$$FRn_{i,min} = -T_{tol}/4 \tag{2}$$

The rotational deviations can be derived geometrically from the kinematic joints by combining several vectors from the loop. The tilting cases with relevant geometric parameters for deriving the equations are shown in Figure 5.

Changes in the tolerance of C4, as defined by the joint configuration, result in a tilting of the rotor R around the contact point at B1 (Point O in Figure 5), caused by moving B2 by $t = T_{C4}$ along the y-axis. Under this assumption, the tilt angle can be calculated:

$$\alpha_{C4} = \arcsin\left(\frac{t}{l_R + r_B}\right) \tag{3}$$

Changes in R4 are a simple rotation of part N on part R around the center of the cylinder axis. This results in a direct influence of the tolerance zone rotation on the final result. The total rotation angle is determined by the characteristic length c_{R4}, which corresponds to the length of the cylindrical surface defined by the geometry of part R for the runout tolerance R4 (see Figure 3):

$$\alpha_{R4} = \arctan\left(\frac{T_{R4}}{c_{R4}}\right) \tag{4}$$

DOI
10.3384/ecp218765

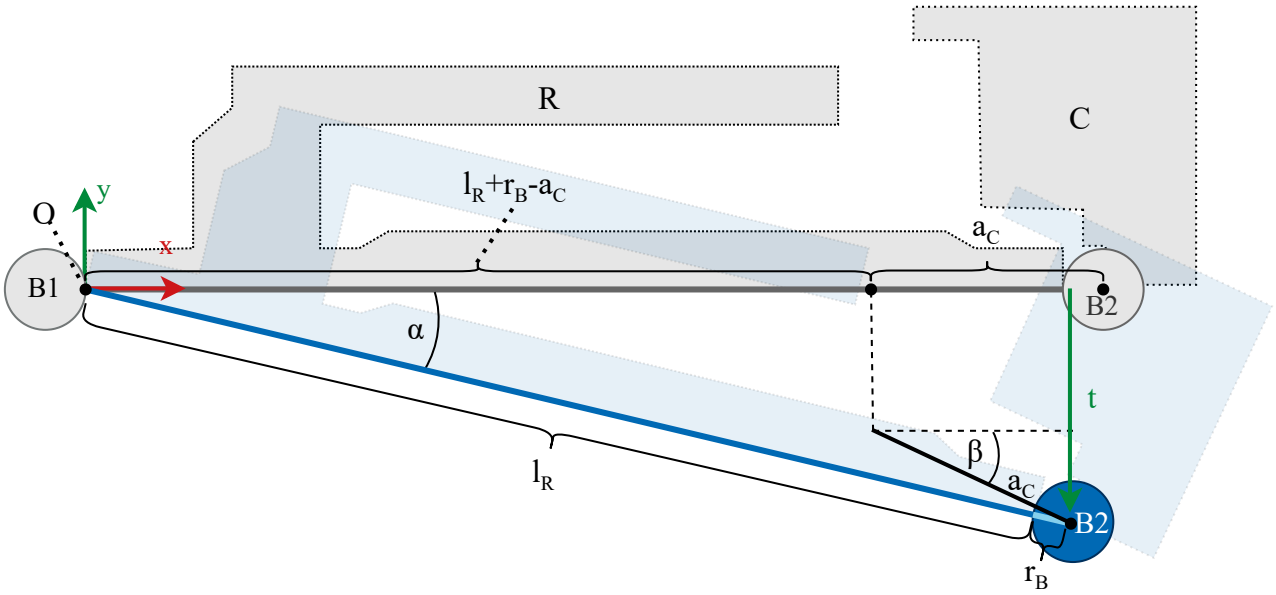

Figure 5. Geometric relationships for calculating the tilt angle α of the rotor between the bearings B1 and B2

Changes in the tolerance of S4 result in a tilting of the rotor R around the contact point at O at Ball B1, as well as a tilting of the cap C around the ball B2. During these movements, contact between the cap C and the stator S in the x-direction is maintained (with the contact defined at a fixed distance a_C). The tolerance variation affects the angle β in this case, which is calculated by using the tolerance zone width T_{S4}: $\beta_{S4} = \arctan(T_{S4}/c_{S4})$. The characteristic feature length c_{S4} can be derived from the geometry of part S. S4 defines the tolerance of the outer cylindrical surfaces shown in Figure 3. Corresponding to Figure 5, α_{S4} can then be determined using this equation:

$$\alpha_{S4} = \arccos\left(\frac{a_C \cdot \cos(\beta_{S4}) + l_R + r_B - a_C}{l_R + r_B}\right) \quad (5)$$

The tilt angles α can be used to determine the change of FRn_i in relation to these measuring points via the position vector of the FRn. For $FR2$, it should be noted that in the case of S4 the reference point for measurement in part C is also rotated by the angle β_{S4}.

Finally, the partial contributions determined in this way are summed up:

$$FRn_{max/min} = FRn_{nominal} + \sum FRn_{i,max/min} \quad (6)$$

The values of FRn_{max} and FRn_{min} calculated in this way form the output of the FMU for this example. The equations are transferred to a Modelica model (Appendix Listing 1) and subsequently exported as an FMU using OpenModelica.

4.3 Results and Verification

A comparison of the results calculated with the FMU and CETOL is shown in Figure 6. The worst-case calculation is performed directly within the FMU based on a predefined set of model inputs, which correspond to those used in the CETOL analysis. The values calculated by the method described above are very close to the result of CETOL with deviations from the worst case values of less than 3%. These minor deviations result from the fact that the assumptions made in this paper regarding the kinematic chain do not exactly reproduce the specific behaviour of the numerical solver used for the joint configuration in CETOL. This FMU can now be integrated into a system simulation to calculate the two FRs and the influence of the mechanical tolerances can be analysed across domains.

To further demonstrate the applicability of the FMU and at the same time present a complementary verification method for the results, additional testing was carried out using FMUGym as a feasible tool for this purpose. The FMU implementation presented above calculates only worst-case values and does not perform any statistical analysis internally. Therefore, a Monte Carlo simulation with 1000 samples was conducted with FMUGym to compare the statistical results with those from CETOL. As in the reference analysis, a process capability index (Cpk) of 1.0 (equivalent to $\pm 3\sigma$ limits) was assumed for all input variables, with random variation around the mean value following a normal distribution.

The results of this Monte Carlo simulation are also plotted in Figure 6. It can be seen that the statistical results from the FMU simulation also match the reference values from CETOL very well. While CETOL computes statistical parameters and sensitivities using the method of moments, based on first-order approximations derived from the partial derivatives with respect to individual variables, our approach leverages Monte Carlo simulation, which may lead to slight differences in the results.

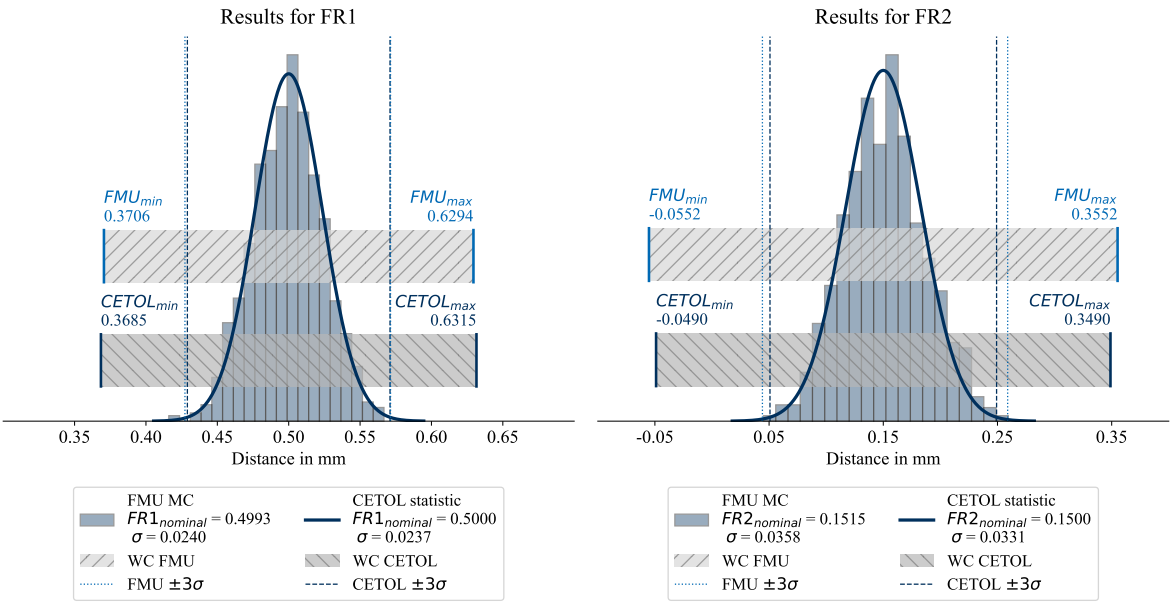

Figure 6. Results of FMU for Worst Case (WC) analysis and Monte Carlo Simulation (MC) as well as the results of the analysis with CETOL. It can be seen that both WC and statistical results of the two methods are close to each other. The deviations result from the assumptions regarding the kinematic chain and the different approaches to determine the statistical results.

5 Discussion

5.1 Evaluation of the Implemented Approach

In the presented case study, the FMU was successfully used to perform a Monte Carlo simulation via the external tool FMUGym. While this demonstrates one specific application scenario, the general concept of using FMUs for tolerance analysis is not limited to this context. All functionalities of FMUs - ranging from simple parameter sweeps to integration in full system simulations - remain applicable.

The FMU developed in this work focuses solely on computing geometric quantities related to the defined FRs. However, in a broader system simulation, these results could be interpreted and used across domains. For example, as described in the case study, the air gap ($FR1$) influences torque generation in the magnetic circuit of the stepper motor, while the guide clearance ($FR2$) can introduce friction forces at the mechanical output. These interactions imply how geometrical tolerances can have a measurable impact on system-level behaviour and performance.

5.2 Usefulness of the FMU Approach

The use of FMUs enables seamless integration of tolerance analysis into complex system models, facilitating cross-domain simulation workflows. This modularization allows tools like FMUGym or Modelica-based environments to interact with geometrically defined tolerances without reproducing detailed internal logic on the system level. The FMU outputs can thus be interpreted either in purely geometrical terms or as system-relevant input variables, depending on the simulation context.

The benefits of standardized, automatable interfaces for solving geometry-related problems in 3D system models has also been supported by recent literature in the field of MBSE (Lehner et al. 2025).

In the future, FMUs offer significant potential for extended applications, such as dynamic simulations with time-dependent input and output variables. This could include non-rigid components or evolving relationships between assembly parts, enabling the modelling of effects like deformation under load, wear, or reconfiguration of mechanical constraints during operation. In the long term, the FMU approach may also allow internal processes and models to be manipulated externally for example, by selecting tolerance types, defining contact graphs, or detecting contact zones. These advanced capabilities, however, are highly dependent on the specific analysis method used and would need to be evaluated individually for each application.

5.3 Limitations

The approach demonstrated in this case study is subject to several limitations that currently restrict its broader applicability. Most notably, the FMU represents a static analysis of rigid-body behaviour and does not account for dynamic effects, deformation, or time-dependent interactions. As such, the FMU output reflects only a snapshot of the geometric relationships under worst-case or statistical assumptions. Without dynamic analyses or automated FMU exports, the approach therefore currently only involves additional implementation effort.

Moreover, the internal structure of the FMU is tightly coupled to the specific tolerance scheme used during its generation. Any modification to the tolerancing concept

DOI
10.3384/ecp218765

- such as changes in the type or arrangement of tolerance elements, beyond the mere numerical values - requires the FMU to be re-generated manually. This limits flexibility, particularly in early design phases where such schemes are subject to frequent change.

Based on the results presented in this paper, a fully automated export of FMUs is not yet feasible. Possible approaches to enable such automation could include systematically translating the methods from subsection 2.1 into DAE systems or representing their behavior using response surfaces as meta-models. A key prerequisite for enabling such automation would be the availability of freely accessible toolchains for geometric tolerance analysis. To our knowledge, the only freely available tool in this domain is *politopix* (Delos and Teissandier 2015). However, since politopix is based on Deviation Models, integrating it into an automated FMU generation workflow would be significantly more complex than with methods such as the Vector Loop approach. Nevertheless, its open accessibility could make it a valuable starting point for future developments in this direction.

6 Conclusion and Outlook

This work introduced a structured concept for integrating geometric tolerance analysis into system simulations using FMUs. A consistent interface based on the FMI standard was proposed, enabling the encapsulation of tolerance logic in a modular and reusable format. Through a case study, the practical feasibility of implementing a manually derived FMU was demonstrated and successfully verified against a commercial analysis tool.

While the current implementation is limited to static, rigid-body models and requires manual derivation of code, it shows that FMUs can represent mechanical tolerances with sufficient precision for system-level applications. With future development, including automation of model generation and support for dynamic or deformable systems, FMUs could serve as a practical bridge between detailed mechanical analyses and multi-domain system simulations.

In the long term, enabling FMU export directly from commercial CAT tools would be a desirable step to facilitate a broader application of geometric tolerance analysis within MBSE processes.

References

Anselmetti, Bernard (2006). "Generation of Functional Tolerancing Based on Positioning Features". In: *Computer-Aided Design* 38.8, pp. 902–919. DOI: 10.1016/j.cad.2006.05.005.

Blochwitz, Torsten et al. (2011). "The Functional Mockup Interface for Tool Independent Exchange of Simulation Models". In: *Proceedings of the 8th int. Modelica Conf.* Linköping University Press, pp. 105–114. DOI: 10.3384/ecp11063105.

Cao, Yanlong, Ting Liu, and Jiangxin Yang (2018). "A Comprehensive Review of Tolerance Analysis Models". In: *The Int. Journal of Advanced Manufacturing Technology* 97, pp. 3055–3085. DOI: 10.1007/s00170-018-1920-2.

Chase, Kenneth W. et al. (1996). "Including Geometric Feature Variations in Tolerance Analysis of Mechanical Assemblies". In: *IIE Transactions* 28.10, pp. 795–807. DOI: 10.1080/15458830.1996.11770732.

Chen, Hua et al. (2014). "A Comprehensive Study of Three Dimensional Tolerance Analysis Methods". In: *Computer-Aided Design* 53, pp. 1–13. DOI: 10.1016/j.cad.2014.02.014.

Davidson, Joseph K., Alma Mujezinovic, and Jami J. Shah (2002). "A New Mathematical Model for Geometric Tolerances As Applied to Round Faces". In: *Journal of Mechanical Design* 124, pp. 609–622. DOI: 10.1115/1.1497362.

Delos, Vincent and Denis Teissandier (2015). *PolitoCAT and Politopix*. http://i2m.u-bordeaux.fr/politopix. University of Bordeaux, Talence, France, accessed April 6, 2025.

Desrochers, Alain, Walid Ghie, and Luc Laperrie're (2003). "Application of a Unified Jacobian—Torsor Model for Tolerance Analysis". In: *Journal of Computing and Information Science in Engineering* 3.1, pp. 2–14. DOI: 10.1115/1.1573235.

Desrochers, Alain and Alain Riviere (1997). "A Matrix Approach to the Representation of Tolerance Zones and Clearances". In: *The Int. Journal of Advanced Manufacturing Technology* 13.9, pp. 630–636. DOI: 10.1007/BF01350821.

Gödecke, Andreas et al. (2012). "FEM models in System Simulations using Model Order Reduction and Functional Mockup Interface". In: *Proceedings of the 9th Int. Modelica Conf., Munich, Germany*. Linköping University Electronic Press, pp. 565–570. DOI: 10.3384/ecp12076565.

Gohl, Jesse et al. (2024). "Steady-state Optimization of Modelica Models and Functional Mockup Units with Pyomo". In: *Proceedings of the American Modelica ConferenceStorrs, Connecticut, USA*, pp. 109–118. DOI: 10.3384/ecp207109.

Gupta, S. and J. U. Turner (1993). "Variational Solid Modeling for Tolerance Analysis". In: *IEEE Computer Graphics and Applications* 13.3, pp. 64–74. DOI: 10.1109/38.210493.

Haghighi, Payam et al. (2014). "Automatic Detection and Extraction of Tolerance Stacks in Mechanical Assemblies". In: *Proceedings of the ASME Int. Design Engineering Technical Conf. and Computers and Information in Engineering Conf.* New York, NY: ASME. DOI: 10.1115/DETC2014-35315.

Hallmann, Martin, Benjamin Schleich, and Sandro Wartzack (2020). "From Tolerance Allocation to Tolerance-cost Optimization: A Comprehensive Literature Review". In: *The Int. Journal of Advanced Manufacturing Technology* 107, pp. 4859–4912. DOI: 10.1007/s00170-020-05254-5.

Homri, Lazhar, Denis Teissandier, and Alex Ballu (2015). "Tolerance Analysis by Polytopes: Taking into Account Degrees of Freedom with Cap Half-spaces". In: *Computer-Aided Design* 62, pp. 112–130. DOI: 10.1016/j.cad.2014.11.005.

Husch, Julia and Michael S. J. Walter (2024). "Tolerances in Mechanisms". In: *Research in Tolerancing*. Ed. by Sandro Wartzack. Cham: Springer Nature Switzerland, pp. 65–99. DOI: 10.1007/978-3-031-64225-8_4.

ISO International Organization for Standardization (2011). *ISO 8015:2011; Geometrical product specifications (GPS) — Fundamentals — Concepts, principles and rules*.

Kosec, P., S. Skec, and D. Miler (2020). "A Comparison of the Tolerance Analysis Methods in the Open-loop Assembly". In: *Advances in Production Engineering & Management* 15.1, pp. 44–56. DOI: 10.14743/apem2020.1.348.

Lehner, Daniel et al. (2025). “Model-driven Engineering for Digital Twins: A Systematic Mapping Study”. In: *Software and Systems Modeling*, pp. 1–39. DOI: 10.1007/s10270-025-01264-7.

Moers, Frederik et al. (2025). “Reusable Solution Element Libraries for Accelerated Application of MBSE in Mechanical Product Development”. In: *Forschung im Ingenieurwesen* 89.1, pp. 1–14. DOI: 10.1007/s10010-025-00825-y.

Morse, Edward et al. (2018). “Tolerancing: Managing Uncertainty from Conceptual Design to Final Product”. In: *CIRP Annals* 67.2, pp. 695–717. DOI: 10.1016/j.cirp.2018.05.009.

Otter, Martin et al. (2022). “Towards Modelica Models with Credibility Information”. In: *Electronics* 11.17. ISSN: 2079-9292. DOI: 10.3390/electronics11172728. URL: https://github.com/DLR-SR/Credibility.

Ramnath, Satchit et al. (2018). “Comparative Study of Tolerance Analysis Methods Applied to a Complex Assembly”. In: *Procedia CIRP* 75, pp. 208–213. DOI: 10.1016/j.procir.2018.04.073.

Schleich, Benjamin et al. (2016). “Status and Prospects of Skin Model Shapes for Geometric Variations Management”. In: *Procedia CIRP* 43, pp. 154–159. DOI: 10.1016/j.procir.2016.02.005.

Shah, Jami J. et al. (2007). “Navigating the Tolerance Analysis Maze”. In: *Computer-Aided Design and Applications* 4, pp. 705–718. DOI: 10.1080/16864360.2007.10738404.

Shen, Zhengshu et al. (2005). “A Comparative Study Of Tolerance Analysis Methods”. In: *Journal of Computing and Information Science in Engineering* 5.3, pp. 247–256. DOI: 10.1115/1.1979509.

Walter, Michael S. J. et al. (2021). “Statistical Tolerance Analysis—A Survey on Awareness, Use and Need in German Industry”. In: *Applied Sciences* 11.6, p. 2622. DOI: 10.3390/app11062622.

Wrede, Konstantin et al. (2024). “FMUGym: An Interface for Reinforcement Learning-based Control of Functional Mock-up Units under Uncertainties”. In: *31st Int. Workshop on Intelligent Computing in Engineering, EG-ICE 2024*, pp. 647–656. DOI: 10.35869/Proceedings_EGICE2024. URL: https://github.com/Fraunhofer-IIS/fmugym.

A Appendix

Listing 1. Modelica model with all equations before FMU-export

```
model geom_tol
// removed in- and outputs to save space
// real inputs: R1, S2, C4, S4, C5, R4, N2
// real outputs: FR1min, FR1_max, FR2_min,
    FR2_max
protected
parameter Real Char_L_S4 = 27.5;
parameter Real Char_L_R4 = 17.2679;
parameter Real l_R = 30.5;
parameter Real r_B = 1.0;
parameter Real a_C = 10.0;
parameter Real Pos_FR1_R_x = 13.96;
parameter Real Pos_FR1_R_y = 7.25;
parameter Real Pos_FR2_RN_x = 31.25;
parameter Real Pos_FR2_RN_y = 1.25;
parameter Real Pos_FR2_N_x = 13.21;
parameter Real Pos_FR2_N_y = 1.25;
parameter Real Pos_P_R_x = 31.5;
parameter Real Pos_P_R_y = 0.0;
parameter Real Pos_FR2ref_C_x = -0.25;
parameter Real Pos_FR2ref_C_y = 1.25;
parameter Real FR1_nom = 0.5;
parameter Real FR2_nom = 0.15;
Real alpha_C4;
Real S4_pos;
Boolean S4_negative;
Real beta_S4;
Real alpha_S4;
Real FR1_S4;
Real XM, YM, XB, YB;
Real FR2_S4;
Real alpha_R4;
Real FR2_R4;
equation
// C4 effect
alpha_C4 = asin(C4/2 / (l_R + r_B));
// S4 sign logic
S4_negative = S4 < 0;
S4_pos = if S4_negative then -S4 else S4;
beta_S4 = atan(S4_pos / Char_L_S4);
alpha_S4 = acos((a_C * cos(beta_S4) + l_R +
     r_B - a_C) / (l_R + r_B));
FR1_S4 = (Pos_FR1_R_y*(1 - cos(alpha_S4)) +
     Pos_FR1_R_x*sin(alpha_S4)) * (if
    S4_negative then -1 else 1);
XM = Pos_FR2_RN_x * cos(alpha_S4) -
    Pos_FR2_RN_y * sin(alpha_S4);
YM = Pos_FR2_RN_x * sin(alpha_S4) +
    Pos_FR2_RN_y * cos(alpha_S4);
XB = Pos_P_R_x * cos(alpha_S4) - Pos_P_R_y
    * sin(alpha_S4) + Pos_FR2ref_C_x * cos(
    beta_S4) - Pos_FR2ref_C_y * sin(beta_S4
    );
YB = Pos_P_R_x * sin(alpha_S4) + Pos_P_R_y
    * cos(alpha_S4) + Pos_FR2ref_C_x * sin(
    beta_S4) + Pos_FR2ref_C_y * cos(beta_S4
    );
FR2_S4 = sqrt((XB - XM)^2 + (YB - YM)^2) *
    (if S4_negative then -1 else 1);
// R4 effect
alpha_R4 = atan(R4 / Char_L_R4);
FR2_R4 = Pos_FR2_N_y*(1 - cos(alpha_R4)) +
    Pos_FR2_N_x*sin(alpha_R4);
// Final outputs
FR1_min = FR1_nom - R1/4 - S2/4
- (Pos_FR1_R_y*(1 - cos(alpha_C4)) +
    Pos_FR1_R_x*sin(alpha_C4))
- FR1_S4;
FR1_max = FR1_nom + R1/4 + S2/4
+ (Pos_FR1_R_y*(1 - cos(alpha_C4)) +
    Pos_FR1_R_x*sin(alpha_C4))
+ FR1_S4;
FR2_min = FR2_nom - C5/4 - N2/4
- (Pos_FR2_RN_y*(1 - cos(alpha_C4)) +
    Pos_FR2_RN_x*sin(alpha_C4))
- FR2_S4
- FR2_R4;
FR2_max = FR2_nom + C5/4 + N2/4
+ (Pos_FR2_RN_y*(1 - cos(alpha_C4)) +
    Pos_FR2_RN_x*sin(alpha_C4))
+ FR2_S4
+ FR2_R4;
end geom_tol;
```