# CraneFlow: An Open-Source Material Flow Simulation Framework for Bridge and Gantry Crane Systems

Vincent Betker*, Mathias Kühn, Thorsten Schmidt

[1]Institute of Material Handling and Industrial Engineering, TU Dresden, 01062 Dresden, Germany
*vincent.betker@tu-dresden.de

SNE 36(3), 2026, 115-122, DOI: 10.11128/sne.36.tn.10781
Selected ASIM SPL 2025 Postconf. Publication: 2026-02-15
Received Revised: 2026-05-27; Accepted: 2026-07-20
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

**Abstract.** Accurate simulation of bridge and gantry cranes is essential to predict a crane system's material flow performance. This paper presents an overview of crane system design methods, evaluates the state of the art of crane modelling in material flow simulation software, and introduces CraneFlow, an open-source Python framework specifically tailored for simulating material flows in bridge and gantry crane systems. We identify limitations in the modelling of jerk-limited motion, motion sequences, and multi-crane interactions in current simulation software. CraneFlow addresses these shortcomings while retaining computational efficiency. Through a case study of an automated coil storage yard, we demonstrate that increased motion modelling detail can significantly alter throughput estimates.

## Introduction

Bridge and gantry cranes perform essential transport tasks within various production and logistics systems, especially in steel production, rail and sea terminals, and warehousing [1, 2]. When integrated into continuous production processes, these cranes typically operate at high utilisation rates, making the correct design of these *process cranes* essential for uninterrupted operations. To design such cranes, one has to determine the specifications of the crane, such as its maximum speed or the load-handling equipment used, as well as the number of cranes needed to achieve the desired system performance. The crane system's performance is further influenced by its *behaviour*, such as the motion sequences of the cranes or their control logic, as well as by the production or logistics systems the crane acts in. While extensive sets of standards exist for the technical design of a crane based on given specifications, the determination of the crane specifications themselves is often based on experience, empirical values, or a static analysis of the expected activities of the crane system. These methods can lead to suboptimal designs, particularly in more complex operation environments. Such environments can be modelled using material flow simulation, leading to more robust design decisions. However, existing material flow simulation software often treats cranes as one means of transport among many, modelling a crane's motion and behaviour in a simplified manner and thereby limiting the model accuracy. To address this issue, this paper

- provides an overview of the state of the art in crane system design, including static methods and dynamic simulation approaches,
- systematically evaluates the crane modelling capabilities of major material flow simulation software,
- introduces CraneFlow, an open-source framework that enables detailed modelling of bridge and gantry cranes with jerk-limited motion sequences,
- demonstrates how increased motion modelling detail impacts simulation results and system design decisions in a case study.

The paper is organised as follows. Section 1 examines motion sequences and cycle times in crane systems, while Section 2 briefly introduces the state of the art in crane system design, presenting both static and dynamic approaches. Section 3 evaluates the crane modelling capabilities of existing material flow simulation software.

Section 4 presents CraneFlow, our open-source framework for detailed crane simulation. Section 5 demonstrates the practical application of CraneFlow and its implications in a case study on an automated coil storage yard. Section 6 provides a conclusion.

# 1 Motion Sequences in Crane Systems

Bridge and gantry cranes form a three-axis motion system in which movements along the axes are generated by independent drives. A bridge (or gantry) movement occurs along the x-axis, the trolley movement takes place along the y-axis, and the hoist handles movement along the z-axis (Figure 1). The cycle times of a crane therefore depend on the performance of these drives, which is expressed in their limits on speed, acceleration, deceleration, and jerk. The maximum values of these parameters apply individually to each drive. Motion along each axis can be modelled in various levels of detail, ranging from simple constant velocity assumptions to comprehensive models that include constant acceleration and even more detailed modelling with jerk limits, resulting in dynamic acceleration values. Jerk (the rate of change of acceleration) is particularly important for realistic modelling of heavy cranes, as they require smooth motion to reduce mechanical stress on the crane structure. Additional crane movements may include the opening and closing of the load-handling attachment and, in some crane designs, rotation of the trolley. The cycle times of a crane depend not only on individual axis movements but also on the superimposition of these movements. For safety reasons, some cranes only begin movements along the x- and y-axes once the hoist has been fully raised, leading to sequential motion. Others allow complete or partial superimposition of movements, enabling parallel operation of all drives and thus shorter cycle times. These motion sequences can significantly alter crane performance and should therefore be accurately represented in simulation models. For multi-crane systems, additional complexity arises from the interaction between cranes sharing the same rails and working areas. Collision avoidance strategies and scheduling algorithms can create waiting times that significantly increase transport cycle times, so a crane simulation model needs to incorporate these as well.

# 2 Design of Crane Systems

The goal of crane system design is to describe a crane system that fulfils its transport tasks while adhering to specified key performance indicators. Target values may include, for example, the maximum waiting time for transport goods, crane utilisation rates, or throughput capacity. The resulting specifications of the crane system should include the number of cranes as well as the performance parameters of the cranes with their individual drives and load-handling systems.

## 2.1 Static Design According to VDI 3573

VDI standard 3573 presents a static method for designing crane systems [3], calculating the throughput of a crane system based on the duration of one working cycle. The working cycle is determined based on the travel distance, the maximum values for speed, acceleration, and deceleration of the crane, as well as the durations of additional activities such as load pick-up/set-down, (fine) positioning, and load weight detection. To determine the travel distance or cycle time of a representative crane trip, VDI 3573 refers to VDI 4446, which provides three different methods for this purpose [4]. The first method is applicable when the cranes' transports have fixed start and destination positions. The second method defines the cycle time calculation for completely filling and emptying a storage area. The third calculation method addresses working cycles for storage areas with stochastically changing fill levels. Here, stochastic inbound and outbound movements are taken into account, which continuously change the contents of the storage area. For the cycle time calculation, two representative points in the storage area for inbound and outbound movements are defined such that the average cycle time between these

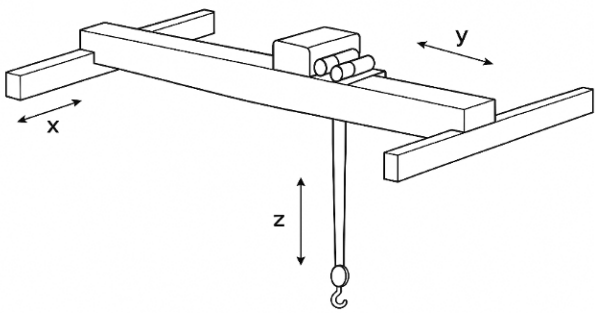



**Figure 1**: Motion axes of a bridge crane.

points corresponds to the mean cycle time of all storage locations. The average system throughput is calculated using the cycle time and the available operating time. While static design methods use detailed cycle time calculations for the representative transports, they have inherent limitations. They cannot account for external dynamic influences such as stochastic variations in request patterns or the impact of failure scenarios. Furthermore, static methods assume deterministic cycle times, whereas real-world operations can exhibit significant variability due to factors such as load-dependent performance, operator behaviour, and environmental conditions. Static crane system design also fails when dealing with cranes operating in a shared workspace.

### 2.2 Dynamic Design Using Simulation

Dynamic design methods come into play when static design methods are not sufficient to describe a crane system [5, p. 5]. Dynamic design takes (stochastic) operational conditions into account, such as arrival times, service times, and failure rates. Material flow simulation is used to model crane systems in those conditions. It provides a model of dynamic processes in order to gain insights that are transferable to reality through simulation experiments [6]. These experiments can be evaluated within a simulation study and allow the observation of a wide range of key metrics, such as waiting times of transport orders or the utilisation of individual crane drives. Simulation models in material flow simulation software can often be visualised, which is used for validation as well as presentation purposes [6].

## 3 Crane Systems in Material Flow Simulation

Material flow simulation models are widely used to model production and logistics systems. Core elements are material handling systems and loads, as well as process resources and buffers. The system behaviour emerges from the interaction of various elements within the simulated environment. In material flow simulation software, crane systems typically appear as part of a material handling system library. However, the implementation details vary between different software. To the best of our knowledge, a systematic survey on crane systems in material flow simulation software has not been published yet. While simulation software evaluation and selection methodologies exist [7, 8, 9], these studies focus on general features of the simulation environment rather than the modelling of specific material handling equipment. Studies using simulation of crane systems can be found in the field of crane scheduling [10], but none offer a comparison of crane modelling in different simulation environments. We therefore fill this research gap by evaluating how crane systems and their movements are modelled in current simulation software. First, we use the work of Dias et al. [11], Lang et al. [12], and Swain [13] to identify widely used material flow simulation software. Then, we assess whether any given simulation software offers support for crane systems based on manuals and marketing material. Last, we evaluate the level of detail in which each software models crane systems by using the software and analysing its documentation.

The evaluation results are presented in Table 1. We list the axes in which the maximum speed ($v_{max}$), maximum acceleration and deceleration ($a_{max}$ and $a_{min}$), and maximum jerk ($j_{max}$) are supported. We distinguish between the movements of the bridge, trolley, and hoist (x-, y-, and z-axis), and further between lifting and lowering (z↑/z↓). If these parameters can be adjusted at runtime, we indicate this using prime notation (x'). We denote motion concurrency options with dashes representing a complete stop of all crane drives: for example, *z-xy-z* means that to move a load, a crane first moves the hoist, then moves along the x and y axes simultaneously before moving the hoist again. Some simulation environments allow a selection of predefined motion sequences, while others allow freely defined sequences. For multi-crane systems, we assess whether the collision avoidance logic can be changed (collision = custom), is fixed (collision = fixed), or is not provided in the software (collision = none). Similarly, crane scheduling can be fixed (scheduling = fixed), customisable (scheduling = custom), or not supplied (scheduling = none).

While all tools simulate bridge and trolley movements with constant acceleration, the level of detail in hoist movement varies. AutoMod and ProModel lack z-axis modelling completely, while AnyLogic models hoist motion with constant velocity. Plant Simulation accounts for acceleration and deceleration along the z-axis, and both Simio and FlexSim further refine this by differentiating between lifting and lowering. However, none of the commercial simulation environments include support for jerk limitations on any axis. Only AnyLogic, Plant Simulation, AutoMod and Visual Components support runtime adjustment of mo-

| | $v_{max}$ | $a_{max}$ | $a_{min}$ | $j_{max}$ | Motion Concurrency | Collision | Scheduling |
|---|---|---|---|---|---|---|---|
| **AnyLogic 8.9.3** | x',y',z' | x',y' | x',y' | - | z-x-y-z, z-xy-z, zxyz | custom | custom |
| **FlexSim 25.0.2** | x,y,z↑,z↓ | x,y,z↑,z↓ | x,y,z↑,z↓ | - | configurable | custom | custom |
| **Plant Simulation 2404** | x',y',z' | x',y',z' | x',y',z' | - | z-xy-z | none | custom |
| **ProModel 10.11.187** | x,y | x,y | x,y | - | xy | fixed | custom |
| **Simio 15.240** | x,y,z↑,z↓ | x,y,z↑,z↓ | x,y,z↑,z↓ | - | z-xy-z | fixed | fixed |
| **AutoMod 14.0.1** | x',y' | x',y' | x',y' | - | xy | none | custom |
| **Enterprise Dynamics 10.6.1** | x,y,z | x,y,z | x,y,z | - | configurable | fixed | custom |
| **Visual Components 4.10** | x',y',z' | x',y',z' | $= -a_{max}$ | - | configurable | none | custom |
| **CraneFlow 1.1** | x',y',z↑',z↓' | x',y',z↑',z↓' | x',y',z↑',z↓' | x',y',z' | configurable | custom | custom |

**Table 1**: Level of detail in crane system modelling across material flow simulation software.

tion parameters. Regarding concurrent axis movement, two groups become apparent: AnyLogic and FlexSim allow for different concurrency patterns, with FlexSim even enabling trajectory-based hook movement. The others allow for a single type of motion sequence with concurrent bridge and trolley movement. All software solutions support multiple cranes on shared rails, but AutoMod and Simio require collision avoidance logic to be provided by the modeller. Of the others with this logic built-in, only FlexSim and AnyLogic allow for its customisation.

# 4 An Open-Source Material Flow Simulation Framework for Crane Systems

Our evaluation shows a limited level of modelling detail of crane systems in existing material flow simulation environments. The absence of jerk-limited motion modelling and restricted customisation options is particularly notable. The closed-source nature of commercially available simulation software further limits changes to the inner workings of the included crane models, reducing the ability to accurately model real-world crane systems. To address this issue, we developed CraneFlow, an open-source framework providing fully customisable crane behaviour. CraneFlow is a specialised material flow simulation environment for bridge and gantry cranes, which accounts for jerk-limited, seven-phase motion on all motion axes. We built CraneFlow on the following principles:

- **Detailed motion modelling:** Support for jerk-limited trajectory planning and multi-crane systems.
- **Configurability:** Extensive options for customising a crane system's behaviour.
- **Performance:** Fast computation times for extensive simulation experiments.
- **Open source:** Fully open-source Python implementation enabling individual adaptations and extensions.

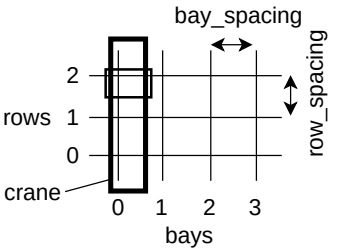
CraneFlow leverages established open-source frameworks by integrating *salabim*'s discrete-event simulation environment [14] with *ruckig* for time-optimal trajectory planning [15], enabling detailed motion modelling with independent velocity, acceleration, deceleration, and jerk constraints for each motion axis. Collision and deadlock avoidance mechanisms for multi-crane systems, developed in our previous work [16], are also integrated. When executed without animation, the framework offers short computation times and is therefore suitable for comprehensive simulation studies, machine learning applications, and real-time environments.

## 4.1 Modules and Features

CraneFlow is structured as a modular framework consisting of several components that work together to model the crane and its operations. In addition, CraneFlow provides basic implementations of material handling equipment to model the loads coming into and leaving the crane system. The main modules used to simulate a crane system are the following:

**Block Module:** The *Block* module represents an area serviced by cranes. This module manages the spatial organisation of stored loads and the management of crane orders. We use the well-established naming convention of *bay*, *row*, and *tier*, with the bays denoting the storage slots along the main movement axis of the crane, the row the positions along the trolley's movement, and the tiers indicating the vertical stacking levels

[17]. With the tiers being a direct result of the height of the stored items themselves, a block can be defined by bays, rows, and the spacing between them (Figure 2). Defining additional pick-up and delivery points outside the grid of bays and rows is possible.




```
from craneflow import Block
block = Block(n_bays=4,
              n_rows=3,
              bay_spacing=6.5,
              row_spacing=2.3)
```

**Figure 2:** Defining a Block.

**Crane Module:** The *Crane* module is the central component of the framework. It models the crane behaviour and implements motion modelling of the bridge (or gantry), the trolley, and the hoist of the crane, supporting independent configuration o f movement parameters for each axis. The parameters defining the individual drives' movement are the limits for velocity (*v_max*, in m/s), acceleration (*acc*, in m/s$^2$), deceleration (*dec*, in m/s$^2$), and jerk, the rate of change of acceleration (*jerk*, in m/s$^3$). The hoist is specified by the same parameters, but these can be set independently for each direction of travel (v_max_up, v_max_down, ...). We can also add waiting times to load or unload the crane (t_connect, t_disconnect), representing the time required for load pick-up/set-down and related operations. It further provides customisable order selection rules for crane scheduling, manages movement concurrency, and implements collision detection and deadlock avoidance for multi-crane systems. This level of detail enables accurate representation of different crane types and sizes. The superimposition of movements in the motion sequence can be freely defined, allowing a partial superimposition to also be modelled. This makes it possible, for example, to implement the start of crane travel upon reaching a safety clearance. A simple example crane can be created as follows:

```
from craneflow import Gantry, Trolley,
Hoist, Crane
crane = Crane(block=block,
              gantry=Gantry(v_max=6.5),
              trolley=Trolley(v_max=2.3),
              hoist=Hoist(v_max_up=1,
                          v_max_down=2,
                          travel_height=6,
                          t_connect=1,
                          t_disconnect=1),
              movement_concurrency='z-xy-z',
              parking_pos=(0, 0))
```

**Load Module:** The *Load* module represents the loads moving through the simulation model. At a minimum, the dimensions of the load need to be defined. A weight can also be added, which can in turn be used to alter crane performance. A load's height defines how far the hoist needs to be lowered to get the load or to place another load on top of it. To move a load, an **Order** is created, which specifies the desired destination of the load:

```
from craneflow import Load, Order
load = Load(length=6, width=2, height=2)
block.add_load(load, stack=(1, 0))
block.add_order(Order(load, to_stack=(3, 2)))
```

**CraneSimulation Module:** The *CraneSimulation* module provides the overarching simulation environment, offering runtime and visualisation control. The simulation model can be run with 2D, 3D, or without visualisation. The visualisation enables visual verification that the crane model behaves as expected, supports communication of simulation results to stakeholders and facilitates the development of the simulation model. A simple simulation model made of the example code snippets produces the visualisation shown in Figure 3 using the following code:

```
from craneflow import CraneSimulation
cs = CraneSimulation(
    components=[block, crane])
cs.run(animate=True)
```

### 4.2 Validation

Given that FlexSim provides the highest level of detail of crane movements among existing tools out-of-the-box, we use it to validate our implementation. To

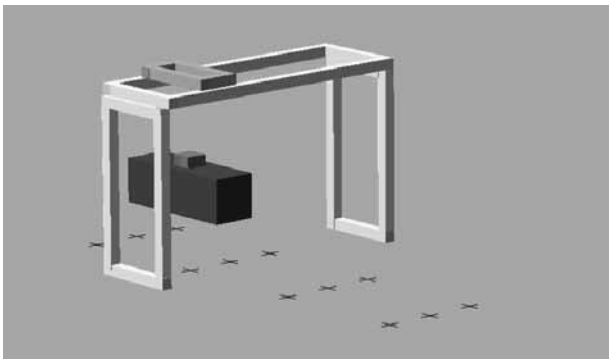


**Figure 3**: visualisation of the example crane.

achieve this, we model a crane with the same specifications i n b oth C raneFlow a nd F lexSim. Working through a given list of jobs, both models show the same behaviour and take the same amount of time to process the tasks. The correct consideration of jerk constraints in the CraneFlow model cannot be validated against FlexSim, as jerk constraints are not available in that software. However, CraneFlow shows consistent results when adding jerk limitations to the model, with motion profiles showing the characteristic seven-phase motion patterns expected from jerk-limited trajectory planning.

# 5 Case Study

We applied our framework in a case study based on a real-world use case to demonstrate the practical application of CraneFlow and to assess the implications of detailed crane motion modelling. The case study covers the design of an automated coil storage yard of a steel mill. CraneFlow was used to simulate the crane system to determine the throughput gains of a second bridge crane.

## 5.1 System Description

The coil yard has a capacity of 880 coils in an area approximately 30 m long and 12 m wide. The incoming coils from the rolling mill are fed to the yard through a conveyor. The coils differ in type and are stored according to a storage plan. Production runs in a three-shift system (24 h/day) with constant output. The outgoing coils are retrieved by trucks, with the coil yard serving as a buffer between the two. The trucks arrive between 6 am and 4 pm and order a type and number of coils according to a given probability distribution. The crane system retrieves the coils from the conveyor, stores them in the yard, and transfers them onto the trucks (Figure 4). The total length of the work area of the crane system from the conveyor to the truck parking spots is approximately 40 m.

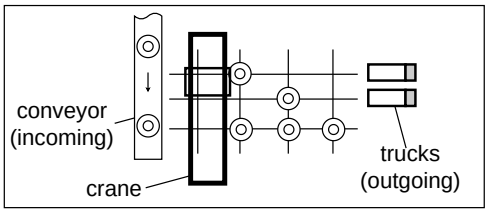



**Figure 4**: Schematic of the coil yard layout.

To measure the maximum throughput of the system in different configurations, we adapted the production input and retrieval processes: for each coil taken off the conveyor by the crane, another one is loaded onto the conveyor. On the retrieval side, a new truck arrives as soon as one leaves. For this experiment, we simplified the scenario further by modelling only one type of coil, and choosing the storage position of the incoming coils as well as the outgoing coils randomly.

## 5.2 Experimental Design

We tested four configurations, differing in the movement constraints of the crane(s):

1. A single crane with velocity, acceleration, and deceleration limits (without jerk limits),
2. A single crane with the same constraints as 1, extended to include jerk limits,
3. Twin cranes with both cranes modelled as in 1,
4. Twin cranes with both cranes modelled as in 2.

The individual crane parameters for each configuration are shown in Table 2.

| Axis | $v_{max}[m/s]$ | $a_{max}, a_{min}[m/s^2]$ | $j_{max}[m/s^3]$ |
|---|---|---|---|
| Bridge | 0.75 | 0.375 | 0.05 |
| Trolley | 0.42 | 0.21 | 0.1 |
| Hoist (Up) | 0.125 | 0.0625 | 0.3 |
| Hoist (Down) | 0.15 | 0.07 | 0.3 |

**Table 2**: Movement constraints of the crane drives.

To evaluate the performance of the different configurations, we ran 100 simulation runs for each configuration with a simulation period of one day. We recorded the number of coils leaving the yard as a measure of system throughput and measured simulation speed to assess computational efficiency. A visualisation of the simulation model is shown in Figure 5.

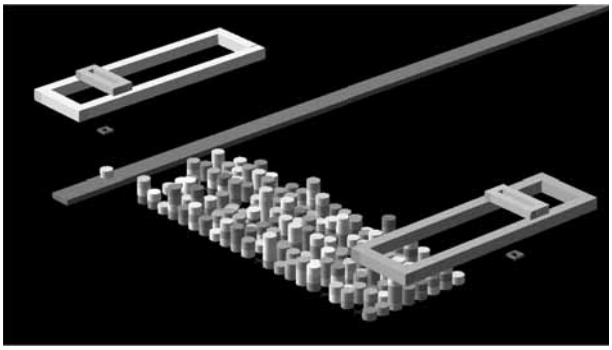


**Figure 5**: visualisation of the coil storage facility with two bridge cranes.

### 5.3 Results

When comparing configurations with and without jerk limits (Table 3), we see a mean throughput decrease from 244 to 229.8 coils for the single crane system ($-5.8$ %), while the throughput of the twin crane system decreases from 468.5 to 437.8 coils on average when introducing jerk limits ($-6.6$ %). This is a significant change that would affect system design decisions. Adding a second crane increases throughput by approximately 92.0 % in the without-jerk configuration and 90.5 % in the with-jerk configuration. This demonstrates the potential of multi-crane systems to increase system throughput, but also shows the limited marginal utility of an additional crane. Regarding computational efficiency, CraneFlow achieves a simulation speed of over 100,000$\times$ relative to real time for the twin crane system, enabling the simulation of one day in roughly one second. The simulation speed for the single crane system was over 170,000$\times$. Using detailed motion modelling with jerk limits did not result in a significant computation time penalty. The results were achieved on a consumer notebook with a CPU clock speed of 4.4 GHz.

The case study demonstrates that the choice of motion modelling detail has a significant impact on simulation outcomes. If a system designer were to base design decisions on a simulation without jerk limits in this use case, they might overestimate the throughput capacity of the intended crane system, potentially leading to undersized systems that cannot meet actual operational requirements.

| Configuration | Coils Out | | Simulation Speed | |
|---|---|---|---|---|
| | mean | SD | mean | SD |
| 1. Single without jerk | 244.0 | 1.1 | 177,870$\times$ | 1,766$\times$ |
| 2. Single with jerk | 229.8 | 1.0 | 176,706$\times$ | 1,911$\times$ |
| 3. Twin without jerk | 468.5 | 3.4 | 105,257$\times$ | 1,170$\times$ |
| 4. Twin with jerk | 437.8 | 3.1 | 104,198$\times$ | 1,594$\times$ |

**Table 3:** Number of outgoing coils and simulation speed for different crane system configurations (SD = standard deviation).

## 6 Conclusion

Material flow simulation is an important tool for the design and evaluation of bridge and gantry crane systems, supporting their reliable and economic operation. However, as shown in this paper, standard material flow simulation software provides limited support for detailed crane motion modelling. While most tools support velocity, acceleration, and deceleration for the bridge and trolley axes, the hoist is often modelled in a simplified way. None support jerk-limited motion on any axis, and customisation options for motion concurrency are restricted. The closed-source nature of commercial simulation software further limits the ability to accurately model crane systems. CraneFlow addresses these limitations of current simulation software by including detailed motion modelling in a discrete-event material flow simulation environment. CraneFlow is a fully open-source Python framework, giving full access to the crane model's internal logic and thus allowing for a closer alignment of crane models with their real-world counterparts. The case study of an automated coil storage yard demonstrates that increased motion modelling detail can significantly alter simulation outcomes, with throughput estimates decreasing by up to 10.1 % when jerk limits are accounted for. This highlights the importance of using accurate motion models when designing heavy crane systems with significant jerk restrictions: if design decisions were based on a simplified crane model, one would risk either oversizing systems and increasing capital costs, or undersizing them and facing performance issues in operation.

### Data Availability

CraneFlow is publicly available at https://tlscm.mw.tu-dresden.de/scm/repo/git/CraneFlow/.

### Publication Remark

This contribution is the revised version of the conference version for ASIM SPL 2025 - 21. ASIM-Fachtagung Simulation in Produktion und Logistik - Dresden 2025, published in Simulation in Produktion und Logistik 2025, ASIM-Mitteilung Nr. 194, ISBN 978-3-86780-806-4, e-ISBN 978-3-86780-809-5, Volume DOI 10.25368/2025.233, Article DOI 10.25368/2025.272

## References

[1] Maschietto GN, Ouazene Y, Ravetti MG, De Souza MC, Yalaoui F. Crane scheduling problem with non-interference constraints in a steel coil distribution centre. *Int. J. Prod. Res.*, 55(6):1607–1622, 2017. doi:10.1080/00207543.2016.1193249.

[2] Guo P, Wang L, Xue C, Wang Y. Dispatching Rules for Scheduling Twin Automated Gantry Cranes in an Automated Railroad Container Terminal. *Arabian J. Sci. Eng.*, 45(3):2205–2217, 2020. doi:10.1007/s13369-019-04176-z.

[3] VDI-Gesellschaft Materialfluss Fördertechnik Logistik. VDI-Richtlinie 3573 Arbeitsgeschwindigkeiten von schienengebundenen Kranen, 2018.

[4] VDI-Gesellschaft Fördertechnik Materialfluss Logistik. VDI-Richtlinie 4446 Spielzeitermittlung von Krananlagen, 2004.

[5] Law AM. Simulation modeling and analysis, Boston: McGraw-Hill; 2007. 768 p.

[6] VDI-Gesellschaft Fördertechnik Materialfluss Logistik. VDI 3633: Simulation of systems in logistics, materials handling, and production, Part 11, 2020.

[7] Nikoukaran J, Hlupic V, Paul RJ. Criteria for simulation software evaluation. In *1998 Winter Simulation Conference*, p. 399–406, Washington, DC, USA. IEEE, 1998. doi:10.1109/WSC.1998.745014.

[8] Tewoldeberhan TW, Verbraeck A, Valentin E, Bardonnet G. An evaluation and selection methodology for discrete-event simulation software. In *2002 Winter Simulation Conference*, p. 67–75, San Diego, CA, USA. IEEE, 2002. doi:10.1109/WSC.2002.1172870.

[9] Fumagalli L, Polenghi A, Negri E, Roda I. Framework for simulation software selection. *J. Simul.*, 13(4):286–303, 2019. doi:10.1080/17477778.2019.1598782.

[10] Boysen N, Briskorn D, Meisel F. A generalized classification scheme for crane scheduling with interference. *Eur. J. Oper. Res.*, 258(1):343–357, 2017. doi:10.1016/j.ejor.2016.08.041.

[11] Dias LMS, Vieira AAC, Pereira GAB, Oliveira JA. Discrete simulation software ranking. In *2016 Winter Simulation Conference*, p.1060–1071, Washington, DC, USA. IEEE, 2016. doi:10.1109/WSC.2016.7822165.

[12] Lang S, Reggelin T, Müller M, Nahhas A. Open-source discrete-event simulation software for applications in production and logistics: An alternative to commercial tools? *Procedia Comput. Sci.*, 180:978–987, 2021. doi:10.1016/j.procs.2021.01.349.

[13] Swain JJ. 2021 Simulation Software Survey Results. https://pubsonline.informs.org/magazine/orms-today/2021-simulation-software-survey, accessed 2026-07-03.

[14] van der Ham R. Salabim: Discrete event simulation and animation in Python. *J. Open Source Software*, 3(27):767, 2018. doi:10.21105/joss.00767.

[15] Berscheid L, Kröger T. Jerk-limited Real-time Trajectory Generation with Arbitrary Target States, 2021. arXiv:2105.04830, doi:10.48550/arXiv.2105.04830.

[16] Betker V, Völker M, Schmidt T. Erarbeitung einer Prozesssteuerungsstrategie für zwei Transportmittel mit gemeinsamen Aktionsbereich am Beispiel eines Prozesskransystems zur Kommissionierung von Schüttgut. In *Tagungsband 19. ASIM-Fachtagung Simulation in Produktion und Logistik*, 2021.

[17] Kemme N. Effects of storage block layout and automated yard crane systems on the performance of seaport container terminals. *OR Spectrum*, 34(3):563–591, 2012. doi:10.1007/s00291-011-0242-7.