DOI: 10.58895/ksp/1000162754-16 erschienen in:
Proceedings. 33. Workshop Computational Intelligence - Berlin, 23. – 24. November 2023
DOI: 10.5445/KSP/1000162754 | https://www.ksp.kit.edu/site/books/e/10.5445/KSP/1000162754/

Modellierung der Raumluftfeuchtigkeit in historischen Gebäuden: Ein Vergleich datengetriebener Methoden in der Praxis

Marcel Zehner, Alessio Cavaterra, Steven Lambeck

FB Elektrotechnik und Informationstechnik, Hochschule Fulda
Leipziger Str. 123, 36037 Fulda
E-Mail: {marcel.zehner,alessio.cavaterra,steven.lambeck}@et.hs-fulda.de

1 Einführung

Bei der Lagerung und Ausstellung von Kulturgütern in historischen Gebäuden spielen die Temperatur und relative Luftfeuchtigkeit eine bedeutende Rolle für deren langfristigen Erhalt. Damit ein beschleunigter Zerfall der Kulturgüter verhindert werden kann, müssen die Raumklimakomponenten gemäß dem Fachgebiet der Präventiven Konservierung (PK) innerhalb vorgegebener konstanter und dynamischer Grenzwertbereiche liegen. Insbesondere der relativen Raumluftfeuchtigkeit und deren Änderungsrate wird eine hohe Relevanz beigemessen [1]. Durch die Implementierung eines modellprädiktiven Regelungsansatzes (MPC, engl.: model predictive control) ist es möglich, die konservatorischen Anforderungen in Form von Regelgrößenbeschränkungen im Regelgesetz zu berücksichtigen. Im Hrabanus-Maurus-Saal (HRMS) innerhalb der Bibliothek des Bischöflichen Priesterseminars in Fulda soll ein solcher modellprädikitver Regelungsansatz realisiert werden. Dort lagern historisch wertvolle Schriften, welche den oben genannten konservatorischen Anforderungen genügen müssen. Bevor der MPC praktisch in der empfindlichen Lagerumgebung umgesetzt werden kann, muss eine simulative Erprobung und Evaluierung des Reglers stattfinden, um einen zuverlässigen Regelbetrieb gewährleisten zu können. Hierfür wird ein Simulationsmodell des HRMS benötigt, welches das hygrothermische Verhalten des Saals mit hoher Genauigkeit

approximiert. Verschiedene Gebäudesimulationsumgebungen, wie beispielsweise EnergyPlus [2] oder TRNSYS [3] erlauben es, das hygrothermische Verhalten eines Gebäudes mitsamt der darin enthaltenen Räume detailgetreu nachzubilden. Für die Nutzung einer solchen Gebäudesimulationsumgebung sind jedoch eine Vielzahl an bauphysikalischen Informationen notwendig. Dazu zählen u. A. die exakten Gebäudeabmessungen, der Wandaufbau oder auch die verwendeten Baustoffe. Da für den HRMS nur sehr wenige bauphysikalische Informationen vorliegen, muss an dieser Stelle ein datengetriebener Modellierungsansatz gewählt werden. Durch langjährige Messungen innerhalb und außerhalb des HRMS steht eine große Datenbasis zur Verfügung. Hiermit ist eine Identifikation eines Simulationsmodells des HRMS möglich, welches im Anschluss für die Reglerevaluierung genutzt werden kann. Der Sachverhalt, dass zusätzlich zum internen Prozessmodell des MPC ein weiteres weitaus detaillierteres Modell des zu regelnden Prozesses notwendig ist, wird in [4] beschrieben. Der vorliegende Beitrag beinhaltet die dynamische Modellierung des hygrischen Verhaltens des HRMS. Auf eine hygrothermische Modellierung des Raums, welche auch das Wärmeübertragungsverhalten einschließt, wird an dieser Stelle verzichtet. In den folgenden Abschnitten werden verschiedene Modellansätze zur datengetriebenen Systemidentifikation vorgestellt. Nach dem Training der Modelle mit Hilfe der Messdaten des HRMS werden diese anschließend miteinander verglichen. Der Beitrag schließt mit einer Diskussion der Ergebnisse ab.

2 Datengetriebene Modellbildung

2.1 Datenbasis

Die Datenbasis erstreckt sich über einen Zeitraum von 140 Tagen vom 20. Dezember 2022 bis zum 8. Mai 2023, wobei die Messdaten mit einer Abtastzeit von 10 Minuten erfasst wurden. Die Daten sind teilweise im geschlossenen Regelkreis und teilweise im offenen Regelkreis aufgezeichnet worden. Im geschlossenen Regelkreis sind Zweipunktregler mit Schwellwerten eingesetzt worden, deren Reglerparameter mehrmals geändert worden sind. Zwischenzeitlich ist der Regelkreis geöffnet worden, um Testsignale in Form von APRB-

Tabelle 1: Übersicht der gemessenen Größen für die Identifikation des Simulationsmodells

	Formelzeichen	Beschreibung
1	ϑ_R	Raumtemperatur
2	ϑ_A	Außentemperatur
3	$\dot{Q}_y$	Heizeingriff im Raum
4	$\dot{Q}_g$	Globalstrahlung
5	$\dot{m}_h$	Be- und Entfeuchtungseingriff im Raum
6	φ_A	rel. Außenluftfeuchtigkeit
7	φ_R	rel. Raumluftfeuchtigkeit

und Chirpsignalen auf den Prozess zu schalten. Die Aufzeichnungen werden in Trainings-, Validierungs- und Testdaten aufgeteilt, wobei die Trainingsdaten 90 % der Gesamtdatenmenge repräsentieren, während jeweils 5 % der Daten für Validierungs- und Testdaten vorgesehen sind. Die Aufteilung der Datensätze hängt von vielen Faktoren ab: beispielsweise müssen die jahreszeitlichen Einflüsse auf das Prozessverhalten sowie Abschnitte mit deutlichen Stellgrößeneingriffen im Trainingsdatensatz enthalten sein, um eine repräsentative Datenbasis zu erreichen. Alle drei Teildatensätze stellen jeweils eine aufeinanderfolgende Zeitreihe dar, um den temporalen Zusammenhang der Daten zu erhalten. Der Testdatensatz beinhaltet den Anfang der Gesamtdatenmenge, während der Validierungsdatensatz am Ende des Gesamtdatensatzes verortet ist. Die übrigen Daten sind Bestandteil des Trainingsdatensatzes.
Im Rahmen der Datenerhebung sind die in der Tabelle 1 aufgelisteten Messgrößen erfasst worden. Im Zuge der Identifikation werden die Messgrößen eins bis sechs als Systemeingänge genutzt, wohingegen die rel. Raumluftfeuchtigkeit φ_R als Systemausgang definiert ist und die zu identifizierende Größe darstellt. Dadurch ergibt sich ein Mehrgrößensystem mit mehreren Eingängen und einem Ausgang (engl. Multi-Input-Single-Output-System, MISO-System). Für das Training der Modelle werden alle Daten auf einen einheitlichen Wertebereich skaliert.

2.2 Takagi-Sugeno-NARX-Modelle

Die gesammelten Daten stellen allesamt Zeitreihen dar, welche mit Hilfe von ARX-Zeitreihenmodellen (engl. autoregressive-model-with-exogenous-input) angenähert können. Im Folgenden wird beispielhaft ein linear-affines SISO-ARX-Modell veranschaulicht.

$$\hat{y}_{\mathrm{ARX}}(k) = \underline{\theta x} = \left(b_1, \ldots, b_{n_b}, -a_1, \ldots, -a_{n_a}, 1\right) \begin{pmatrix} u(k-1) \\ \vdots \\ u(k-n_b) \\ y(k-1) \\ \vdots \\ y(k-n_a) \\ o \end{pmatrix} \tag{1}$$

Aufgrund der zugrundeliegenden Nichtlinearitäten in Wärme- bzw. Feuchteübertragungsvorgängen eignen sich lineare ARX-Modelle gemäß Gleichung 1 nur bedingt, wenn mit dem geschätzten ARX-Modell eine Simulation (unendlicher Vorhersagehorizont) des Prozesses angestrebt wird. Zur Approximation nichtlinearer Prozesse werden deshalb in der einschlägigen Literatur verschiedene Erweiterungen vorgeschlagen, die unter dem Begriff der NARX-Modelle (N für engl. nonlinear) eingruppiert werden. NARX-Modelle erweitern die ARX-Modellstruktur in Gleichung 1 um eine nichtlineare Funktion $f(\cdot)$, wie in der folgenden Gleichung 2 dargestellt wird.

$$\hat{y}_{\mathrm{NARX}}(k) = f(\underline{\theta x}) \, . \tag{2}$$

Takagi-Sugeno (TS) Fuzzy-Modelle beschreiben eine Klasse von nichtlinearen Modellen mit einem unscharfen (engl. fuzzy) Regelwerk, welche in der Konklusion eine oder mehrere scharfe Ausgangsgrößen, z. B. über ARX-Modelle, bilden. Derartige TS Fuzzy-Modelle werden häufig kurz als TS-NARX-Modelle bezeichnet. In diesem Beitrag wird im weiteren Verlauf TSNX als Abkürzung genutzt.

Zur Schätzung eines passenden TSNX-Modells wurde die LMN-Toolbox [5] mitsamt des darin implementierten LOLIMOT-Konstruktionsalgorithmus verwendet.

$$\underline{\hat{y}}_{\mathrm{TSNX}}(k) = \sum_{i=1}^{n_m} \Phi_i(\underline{z})\,\underline{\theta}_{\mathrm{TSNX},i}\underline{x}\,. \tag{3}$$

Die Fuzzy-Basisfunktionen $\Phi_i(\underline{z})$ haben ihren Wertebereich zwischen null und eins. Berechnet werden sie aus dem Mittel der Zugehörigkeitsgrade

$$\Phi_i(\underline{z}) = \frac{\mu_i(\underline{z})}{\sum\limits_{i=1}^{n_m} \mu_i(\underline{z})} \quad , \quad \sum_{i=1}^{n_m} \Phi_i(\underline{z}) = 1\,. \tag{4}$$

Alle $\mu_i(\underline{z})$ Zugehörigkeitsfunktionen (ZF) werden als Gaußglocken definiert. Mit dem Zentrum ν und der Standardabweichung σ gilt für eine Gauss'sche ZF:

$$\mu_{\mathrm{Gauss}}(x) = \exp\left(\frac{-(x-\nu)^2}{2\sigma^2}\right)\,. \tag{5}$$

Die Schedulingvariable $\underline{z}$ ist im vorliegenden Fall ein Vektor, der die Außentemperatur, die Innentemperatur, die relative Luftfeuchtigkeit im Außenbereich und die relative Luftfeuchtigkeit im Innenraum enthält. Diese Größen werden allesamt aus dem letzten Abtastschritt herangezogen.

$$\underline{z} = [\varphi_R(k-1),\ \vartheta_R(k-1),\ \varphi_A(k-1),\ \vartheta_A(k-1)]^T \tag{6}$$

Die Festlegung der Hyperparameter wird durch eine systematische Suche realisiert. Die Anzahl n_a und $n_{b,1},\ldots,n_{b,6}$ der zurückliegenden Regressoren wird variiert, sodass vergangene Einflussgrößen aus einer, zwei, vier, acht bzw. zwölf Stunden in die Modelle einfließen. Bei einer 10-minütigen Abtastung des Datensatzes entspricht dies 6, 12, 24, 48 und 72 Abtastschritte je Einflussgröße. Als Abbruchkriterium für den LOLIMOT-Algorithmus wird die maximale Teilmodellanzahl ausgewählt und im Bereich von 20, 40, 80 und 160 variiert.

Das resultierende TS-Modell besteht aus $n_m = 16$ Teilmodellen mit insgesamt $n_{\mathrm{par}} = 8080$ Parametern und greift auf die Einflussgrößen der letzten zwölf Stunden zurück. Im Zuge des Vergleichs mit den anderen Modellen wird das

TSNX-Modell simuliert, wobei der Modellausgang zurückgeführt wird und eine Output-Error-Struktur (OE) entsteht.

2.3 Local Model State Space Networks

Local Model State Space Networks (LMSSN) repräsentieren eine neue Klasse nichtlinearer Zustandsraummodelle, deren Struktur auf lokalen Modellnetzen (LMN, engl. local model networks) aufbaut [6]. Zur Approximation der Nichtlinearitäten eines Prozesses wird eine ähnliche Herangehensweise basierend auf einem Regelwerk wie bei den zuvor besprochenen TSNX-Modellen gewählt: mehrere Teilmodelle, welche für unterschiedliche Arbeitspunkte, externe Planungsparameter usw. gültig sind, können einen nichtlinearen Prozess besser abbilden, als ein vergleichbares lineares Zustandsraummodell. Die Verknüpfung dieser Teilmodelle zu einem nichtlinearen Gesamtmodell ermöglicht hierdurch große Gültigkeitsbereiche. Die Besonderheit bei LMSSN ist, dass die nichtlinearen Funktionen $f(\cdot)$ und $g(\cdot)$ in

$$\underline{x}(k+1) = f(\underline{x}(k), u(k)) \tag{7}$$

$$y(k) = g(\underline{x}(k), u(k)) \tag{8}$$

auf unterschiedliche Art und Weise angenähert werden können. Einerseits besteht die Möglichkeit jede i-te Zeile der Zustandsgleichung (7) als einzelnes LMN zu betrachten und anschließend für jedes einzelne LMN einen Konstruktionsalgorithmus wie LOLIMOT oder HILOMOT separat zu nutzen. Andererseits kann die gesamte Zustandsgleichung in einem lokalen Modellnetz in Form eines MIMO-LMSSN berücksichtigt und anschließend mit den genannten Algorithmen konstruiert werden. Gleiches gilt für die Ausgangsgleichung der Zustandsraumdarstellung (8). Die LMSSN-Toolbox bietet dem Nutzer zudem die Möglichkeit eine individuelle Auswahl der mit nichtlinearen Phänomenen behafteten Eingangsgrößen und Zustandsgrößen. Somit kann vom Nutzer gezielt apriorisches Wissen in den Algorithmus eingebracht und möglicherweise der Rechenaufwand verringert werden. Das in diesem Beitrag vorgestellte LMSSN zur Modellierung der Feuchteübertragungsmechanismen stellt ein nichtlineares, zeitdiskretes Zustandsraummodell zweiter Ordnung ($n_x = 2$)

mit $n_p = 6$ Eingängen und einem Ausgang $n_q = 1$ dar.

$$\hat{x}_i(k+1) = \sum_{j=1}^{n_{m,i}} \left[o_{i,j} + \underline{a}_{i,j}^T \underline{\hat{x}}(k) + \underline{b}_{i,j}^T \underline{u}(k) \right] \Phi_{i,j}^{[s]}(k) \,, \tag{9}$$

$$\hat{y}_{\mathrm{LMSSN}}(k) = \sum_{m=1}^{n_o} \left[p_m + \underline{c}_m^T \underline{\hat{x}}(k) + \underline{d}_m^T \underline{u}(k) \right] \Phi_m^{[o]}(k) \,. \tag{10}$$

Jede Zeile in dieser Zustandsraumdarstellung ist als eigenständiges LMN definiert, sodass das LMSSN insgesamt aus drei einzelnen LMN besteht. Die erste Zustandsgleichung bzw. das erste LMN wird vom LOLIMOT-Algorithmus in zwei Teilmodelle aufgeteilt: $n_{m,1} = 2$. Die zweite Zustandsgleichung bzw. das zweite LMN besteht nur aus einem Teilmodell, $n_{m,2} = 1$. Die Ausgangsgleichung (drittes LMN) wird mit zwei Teilmodellen abgebildet, $n_o = 2$. Obwohl in der LMSSN-Toolbox alle Einflussgrößen als Kandidaten zur weiteren Aufteilung deklariert werden, hat der Algorithmus für das erste und das dritte LMN nur bezüglich der ersten beiden Zustände aufgeteilt. Das resultierende LMSSN beinhaltet insgesamt 45 Parameter.

2.4 NARX-Netze

NARX-Netze (NXN) basieren auf Gleichung 2. Die nichtlineare Funktion $f(\cdot)$ wird dabei mit Hilfe eines künstlichen neuronalen Netzes (KNN) angenähert. Bei den Neuronen der Zwischenschicht des KNN findet eine Zündung mit der $f_{\tanh}(\cdot)$ - Aktivierungsfunktion statt. Die Parametermatrix $\underline{\theta}_{\mathrm{NXN}}$ gewichtet die Regressoren $\underline{x}$. $\underline{W}_{\mathrm{out}}$ sorgt für eine weitere Gewichtung und stellt die eigentlichen Gewichte des KNN dar. Zusätlich dazu wird ein Offsetvektor $\underline{\beta}_{\mathrm{out}}$ addiert. Dadurch ergibt sich für die erste Zwischenschicht des NARX-Netzes folgende Gleichung:

$$\hat{y}_{\mathrm{NXN}}(k) = \underline{W}_{\mathrm{out}} f_{\tanh} \left(\underline{\theta}_{\mathrm{NXN}} \underline{x} + \underline{\beta}_{\mathrm{out}} \right) \,. \tag{11}$$

Im Verlauf des Trainingsprozesses werden neben der Anzahl der Regressoren, beispielsweise auch die Anzahl der Neuronen variiert. Das daraus folgende Modell hat eine Zwischenschicht mit 10 Neuronen und die Regressoranzahl sowohl für die Eingänge als auch den Ausgang liegt bei 24. Bei einer Abtastzeit von 10 Minuten entspricht dies einem Zeitraum von 4 Stunden. Für das

Training wird der Levenberg-Marquardt-Algorithmus genutzt. Bei genannter Konfiguration liegt eine Parameteranzahl von $n_{\text{par}} = 1701$ vor. Für den späteren Vergleich liegt das Modell im Gegensatz zum Trainingsprozess nicht mehr in der vorgestellten NARX-Modellstruktur vor. Dies liegt daran, dass für die Simulation der Modellausgang zurückgekoppelt werden muss. Dadurch ergibt sich eine OE-Struktur. Die Implementierung und das Training der NARX-Netze findet mit der Deep Learning Toolbox von MATLAB statt.

2.5 Neural State Space Models

Die grundlegende Formulierung von Neural State Space Models (NSSM) ist in den Gleichungen (12) und (13) zu erkennen. Hierbei sind f_{nss} und g_{nss} KNNs und $x(k)$ ist der Zustand des Systems zum Zeitpunkt k. Die Eingangsgrößen des Systems zum Zeitpunkt k werden durch $u(k)$ repräsentiert. Bei p handelt es sich um die Gewichte des KNN, die im Laufe des Trainingsprozesses identifiziert werden müssen.

$$\underline{x}(k+1) = f_{\text{nss}}\left(\underline{x}(k), \underline{u}(k), \underline{p}\right) \tag{12}$$

$$y(k) = g_{\text{nss}}\left(\underline{x}(k), \underline{p}\right) \tag{13}$$

Bei einer beispielhaften Betrachtung der ersten Zwischenschicht eines NSSM ergeben sich die Gleichungen 14 und 15. Dabei stellen $\underline{V}_{\text{A}}$, $\underline{V}_{\text{B}}$ und $\underline{V}_{\text{C}}$ die Gewichtung der Zustände und Eingänge dar. Zusätzlich dazu und ähnlich wie in Gleichung 11 gibt es die zusätzlichen Gewichtungen $\underline{W}_{\text{AB}}$ und $\underline{W}_{\text{C}}$. Weitere Parameter, die während des Trainingsprozesses identifiziert werden müssen, sind die zusätzlichen Offsetvektoren β_{AB} sowie β_{C} [7].

$$\hat{x}(k+1) = \underline{W}_{\text{AB}} f_{\tanh}\left(\underline{V}_{\text{A}}\hat{\underline{x}}(k) + \underline{V}_{\text{B}}\underline{u}(k) + \underline{\beta}_{\text{AB}}\right) \tag{14}$$

$$\hat{y}_{\text{NSSM}}(k) = \underline{W}_{\text{C}} f_{\tanh}\left(\underline{V}_{\text{C}}\hat{\underline{x}}(k) + \underline{\beta}_{\text{C}}\right) \tag{15}$$

Im Rahmen einer Hyperparameterstudie des NSSM werden beispielsweise die Anzahl der Zwischenschichten, die Anzahl der Neuronen pro Schicht oder auch die Größe der Batches variiert. Auch die Anzahl der Zustände wird im

Laufe der Studie mitberücksichtigt. Das aus der Hyperparameterstudie resultierende Modell erster Ordnung besitzt 2 Zwischenschichten mit je 16 Neuronen und wird während des Trainings mit einer Batchgröße von 1024 Datenpunkten gespeist. Für das Training wird der Adam-Optimierer genutzt. Das resultierende Modell weist zudem eine Parameteranzahl $n_{\mathrm{par}} = 417$ auf, was den Gewichten im Zustandsnetzwerk entspricht. Auf das Training eines Ausgangsnetzwerkes wird an dieser Stelle verzichtet, wodurch g_{nss} der Einheitsmatrix I entspricht. Die Umsetzung und Implementierung der NSSM erfolgt mit Hilfe der Deep Learning Toolbox und der System Identification Toolbox von MATLAB.

3 Vergleich der Ergebnisse

Die vorgestellten Modelle werden auf den Testdaten des HRMS evaluiert und anschließend miteinander verglichen. Diese erstrecken sich über einen Zeitraum von 9 Tagen im Dezember 2022. Für den Vergleich der Modelle werden unterschiedliche Fehler- und Gütemaße herangezogen. Neben dem root mean squared error (RMSE) und dem sum of squared errors (SSE) wird auch der variance accounting for (VAF) für die Beurteilung des Modells genutzt. Bei dem RMSE und dem SSE handelt es sich um Fehlermaße. Im Gegensatz dazu ist der VAF ein Gütemaß. Mit dem RMSE und dem SSE kann grundlegend die Abweichung zwischen Messdaten und Modellausgang evaluiert werden. Mit dem VAF kann untersucht werden, mit welcher Genauigkeit das Modell vor Allem die Dynamik der Messdaten abbildet, weil Offsetfehler keinerlei Einfluss auf das VAF-Gütemaß haben. Für eine detaillierte Erläuterung der Fehler- und Gütemaße wird auf [8] verwiesen. In Tabelle 2 sind die Fehler- und Gütemaße für alle Modelle aufgelistet. Außerdem zeigt Tabelle 2 die Anzahl der Modellparameter zur Beurteilung der Modellkomplexität. Es ist ersichtlich, dass sowohl der RMSE als auch der SSE für das NXN-Modell, LMSSN-Modell und TSNX-Modell nahe beieinander liegen. Im Gegensatz dazu kann das NSSM-Modell den geringsten RMSE und SSE aufweisen. Das TSNX-Modell besitzt den höchsten VAF und bildet die Dynamik der Daten mit hoher Genauigkeit ab. Eine graphische Gegenüberstellung der gemessenen Daten und der Ausgänge der unterschiedlichen Modellstrukturen auf den Testdaten ist in

Tabelle 2: Übersicht der Fehler- und Gütemaße sowie die Anzahl der Parameter aller Modelle

	RMSE [%rH]	SSE [%rH^2]	VAF [%]	n_{par}
NSSM	0.68	585.5	87.07	417
NXN	1.05	1370.1	82.57	1701
LMSSN	1.02	1318.8	83.21	45
TSNX	1.06	1413.1	94.65	8080

Bild 1 zu finden. Hier sind neben den Modellausgängen und den dazugehörigen Fehlerzeitreihen, auch die skalierten Eingänge des Modells sowie jeweils ein Geigenplot aller Modelle zur Beurteilung der Fehlerverteilung erkennbar. Der im Geigenplot mit einer etwas dunkleren Farbgebung dargestellte Bereich repräsentiert den Interquartilsabstand, welcher 50 % aller Daten beinhaltet. Grundsätzlich weisen alle Modelle gute Approximationseigenschaften auf und können die Messdaten unter Berücksichtigung der Dynamik annähern. Bei Betrachtung des zeitlichen Verlaufs des Fehlers oder auch des Geigenplots fällt auf, dass sich der Fehler ungefähr im Bereich $\pm$ 2 %rH aufhält, was den Toleranzen typischer Feuchtesensoren entspricht. Das NSSM-Modell weist die geringste Abweichung auf. Dies ist nicht nur im Fehlerverlauf erkennbar, sondern auch im Geigenplot des NSSM-Modell. Hier liegt der Median (weißer Punkt im Geigenplot) der Fehlerverteilung bei annähernd null. Bei Betrachtung des NXN-Modells und LMSSN-Modells ist eine Verschiebung der Fehlerverteilung in den positiven Wertebereich ersichtlich. Dies bedeutet, dass die Modelle die Testdaten zu einem Großteil der Zeit unterschätzen. Das TSNX-Modell bringt eine noch größere Abweichung mit sich, ist jedoch mit der kleinsten Varianz gekennzeichnet. Das TSNX-Modell weist zudem die höchste Modellkomplexität mit $n_{\text{par}} = 8080$ Parametern auf. Dazwischen liegen das NXN-Modell mit 1701 Parametern und das NSSM-Modell mit einer Parameteranzahl von 417. Die geringste Modellkomplexität besitzt das LMSSN-Modell mit 45 zu schätzenden Parametern.

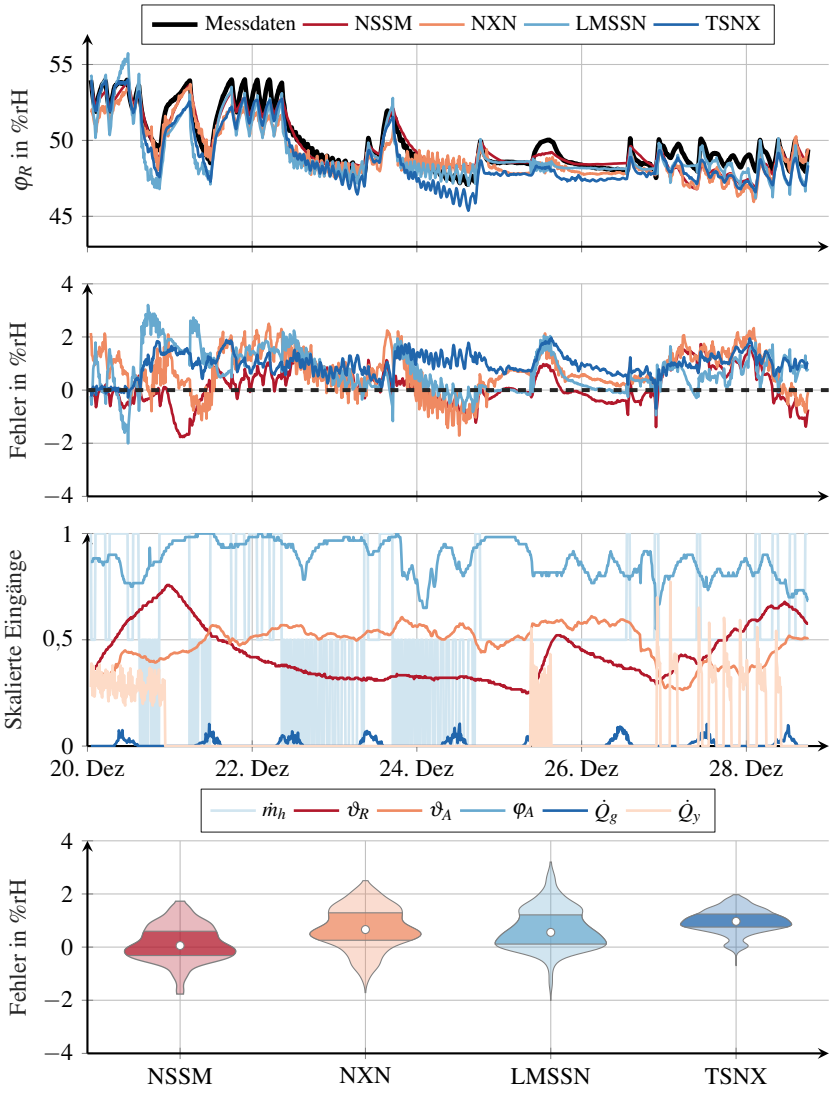

Bild 1: Überblick über die gemessenen Feuchtedaten, die Modellausgänge, die Fehlerzeitreihen, die skalierten Eingangsgrößen der Modellstrukturen, wobei für $\dot{m}_h$ ein Wert > 0.5 eine Befeuchtung und ein Wert kleiner < 0.5 eine Entfeuchtung darstellt, sowie einen Geigenplot zur Beurteilung der Fehlerverteilung zwischen gemessenen Daten und Modell

4 Diskussion und Ausblick

Die Erhebung von aussagekräftigen Messdaten gestaltet sich aufgrund der Notwendigkeit der Einhaltung der konservatorischen Anforderungen an die Kulturgüter schwierig: schnelle Änderungen der Raumklimaparameter sind zu vermeiden, sodass die Anwendung von Testsignalen im Prozess mit äußerster Vorsicht und unter Berücksichtigung weiterer Kriterien, wie z. B. den Außenbedingungen, vorzunehmen sind. Unter Umständen können so wichtige Arbeitspunkte bzw. Arbeitsbereiche des Prozesses nur unzureichend abgedeckt werden. Hinzu kommt, dass ein Teil der Messdaten im geschlossenen Regelkreis aufgezeichnet worden sind, wodurch eine Korrelation der Messgrößen zustande kommt, welche sich negativ auf eine erfolgreiche Systemidentifikation auswirken können. Der interessierte Leser wird auf die Identifikationsbedingungen von [9] verwiesen. Im Laufe des Beitrags wird deutlich, dass die untersuchten Modellstrukturen trotz der genannten Nachteile im Messdatensatz grundsätzlich gute Approximationsgüten aufweisen. Jedes Modell hat unterschiedliche Vorteile und Nachteile, wobei drei von vier Modellen einen Offsetfehler aufzeigen. Dieser Fehler resultiert unter Umständen aufgrund der Korrelation zwischen Eingangs- und Ausgangsgrößen, wodurch die verschiedenen Schätzmethoden keine konsistenten Ergebnisse liefern. Das TSNX-Modell besticht mit einer sehr geringen Varianz in der Fehlerverteilung, bringt aber mit 8080 Parametern die größte Modellkomplexität mit sich. Im Gegensatz dazu, hat das LMSSN-Modell die geringste Modellkomplexität aller Modelle, zeigt jedoch die ausgeprägteste Fehlerstreuung unter den untersuchten Modellen. Ein weiterer Nachteil bei dem LMSSN-Modell ist die lange Trainingsdauer (48 Stunden) im Vergleich zu den anderen Modellen. Mit Blick auf die Nutzung als Simulationsmodell für die Evaluierung modellprädiktiver Regelungsansätze zeigt das NSSM-Modell Potential, da es nicht nur die geringste Abweichung zu den Messdaten zeigt, sondern auch in Kombination mit der Model Predictive Control Toolbox von MATLAB ohne große Portierungsmaßnahmen genutzt werden kann. In künftigen Arbeiten stehen weitere datengetriebene Methoden, wie z. B. physikgeführte neuronale Netze im Fokus.

Wir möchten dem Team von Prof. Dr.-Ing. Oliver Nelles für die Bereitstellung der LMSSN-Toolbox sehr herzlich danken.

Literatur

[1] A. Burmester. „Die Beteiligung des Nutzers bei Museumsneubau und -sanierung: Oder welche Klimawerte sind die Richtigen?“ In: *Raumklima in Museen und historischen Gebäuden*. S. 9–24. 2000

[2] D. Crawley, L. Lawrie, O. Pederseb und F.C. Winkelmann. „EnergyPlus: Energy simulation program“. *Ashrae Journal*. 42, S. 49–56. 2000

[3] S.A. Klein. „TRNSYS-A Transient System Simulation Program“. *University of Wisconsin-Madison*. 1988

[4] G. Serale, M. Fiorentini, A. Capozzoli, D. Bernardini und A. Bemporad. „Model Predictive Control (MPC) for Enhancing Building and HVAC System Energy Efficiency: Problem Formulation, Applications and Oppurtunities“. *Energies*. 11. 2018

[5] B. Hartmann, T. Ebert, T. Fischer, T.Belz, J. Kampmann und O. Nelles. „LMNtool – Toolbox zum automatischen Trainieren lokaler Modellnetze“. *22. Workshop Computational Intelligence, KIT Scientific Publishing*. 45, S. 341–355. 2012.

[6] M. Schüssler. „Machine Learning with nonlinear state space models“. Dissertation. *Universität Siegen. Institut für Mechanik und Regelungstechnik – Mechatronik* 2022

[7] J. A. K. Suykens, B. L. R. De Moor und J. Vandewalle „Nonlinear system identification using neural state space models, applicable to robust control design“. *International Journal of Control*. Vol. 62., S. 129–152. 1995

[8] A. Kroll und H. Schulte. „Benchmark problems for nonlinear system identification and control using Soft Computing methods: Need and Overview“. *Applied Soft Computing*. 22, S. 496–513. 2014

[9] R. Isermann und M. Münchhof. „Parameter Estimation in Closed Loop“. In: *Identification of Dynamic Systems: An Introduction with Applications Springer*. 2011