

Entwurf und Implementierung komplexer Prozessautomatisierungsstrategien auf einer SPS

Von Parasar Kodati, Tom Erkinen, und [Arkadiy Turevskiy](#)

Speicherprogrammierbare Steuerungen (SPS) werden zur Steuerung und Regelung von Prozessen eingesetzt, die von relativ einfachen SISO-Systemen bis hin zu Systemen mit mehreren gekoppelten Regelkreisen und komplexen Überwachungsalgorithmen reichen. Bei einfachen Regelungsaufgaben, die z.B. mit PID-Reglern auskommen, kann man den Regler noch einfach implementieren und seine Parametrierung während des Betriebes der Anlage optimieren. Je höher jedoch die Komplexität der Steuerungs- bzw. Regelungsanwendung wird, desto anspruchsvoller wird auch die Codegenerierung und Verifikation der Steuerlogik von SPS-Systemen. Die Entwickler müssen die Werte zahlreicher Parameter ermitteln und sicherstellen, dass alle Teile des Algorithmus wie vorgesehen zusammenarbeiten. Die Abstimmung eines komplexen Reglers auf einem Hardwareprototypen oder im laufenden Prozess ist zudem nicht nur zeitraubend, sondern es besteht auch ein beträchtliches Risiko zur Schädigung der Anlage.

Die Lösung dieses Problems besteht darin, komplexe Steuerungs- und Regelungsstrategien als Modelle zu entwerfen, die deren Simulation und Verifikation gestatten. Dasselbe Modell kann anschließend zur automatischen Generierung von IEC 61131-konformem strukturierten Text und somit zur Programmierung der SPS verwendet werden. Wie dieser Ansatz in der Praxis gehandhabt wird, stellt dieser Artikel am Beispiel eines Stahlwerks vor.

Das Stahlwalzwerk: Anforderungen an den Reglerentwurf

Ein Walzwerk erzeugt aus einer Bramme ein Stahlblech mit einheitlicher Dicke. Dies geschieht typischerweise in mehreren Walzstufen, bei deren Durchlaufen die Bramme sukzessiv an Dicke verliert, bis sie als Blech die Walzstraße verlässt. Durch zwischenliegende Schlingenheber wird das Blech unter Zugspannung gehalten und ein Abreißen oder Durchhängen verhindert.



Abb. 1: Schema des Walzverfahrens.

Um den mehrstufigen Prozess zu simulieren, modelliert und regelt man zunächst nur eine einzelne Walzstufe. Die gesamte Walzstraße lässt sich anschließend durch Zusammenschaltung mehrerer solcher Einzelstufen aufbauen und analysieren.

Das Regelungssystem des einfachen Walzwerks muss die folgenden Anforderungen erfüllen:

- Der erzeugte Stahl muss nach der letzten Walze eine Dicke von konstant 8 mm \pm 0,1 mm aufweisen.
- Das Stahlblech muss nach der letzten Walze mit einer Geschwindigkeit von 1 m/s \pm 0,1 m/s transportiert werden.
- Die Spannung muss an jeder Walze nach 100 Sekunden 175 kN/m² betragen.
- Fehler in Sensoren und Aktuatoren müssen detektiert und entweder automatisch korrigiert werden oder die Anlage muss sicher heruntergefahren werden.

Modellierung der Anlage als Regelstrecke

Im ersten Schritt wird ein Simulink[®]-Modell des Walzwerks erzeugt, auf dessen Basis anschließend der Regler entwickelt und getestet wird. Der Walzprozess besteht im Wesentlichen aus zwei Komponenten, den eigentlichen Walzen zur Formung des Bleches sowie den jeweils dazwischen liegenden Schlingenhebern. In der Walzstufe bringt ein hydraulischer Aktuator die Walzdruckkraft auf das Stahlblech auf. Das durch den Motorantrieb erzeugte Walzmoment wird zur Regelung der Walzgeschwindigkeit verwendet. Die mechanischen, elektrischen und hydraulischen Komponenten der Walze können mit SimMechanics[™], Simscape[®] und SimHydraulics[™] ohne explizite Herleitung der benötigten Bewegungsgleichungen modelliert werden.

Die Modellierung der Schlingenheber findet ausschließlich in SimMechanics statt. Der Heber sowie das davor und dahinter befindliche Stahlband werden als drei durch Gelenke miteinander verbundene Elemente dargestellt. Abschließend werden die Walzstufen- und die Schlingenheber-Modelle zu einem einzigen Simulink-Modell zusammengefügt.

Entwurf und Verifikation der Regler

Im nächsten Schritt dient das Anlagenmodell für den Entwurf der Regler. Abbildung 2 zeigt die aus mehreren Schleifen bestehende Architektur eines typischen Regelungssystems für ein mehrstufiges Walzverfahren.

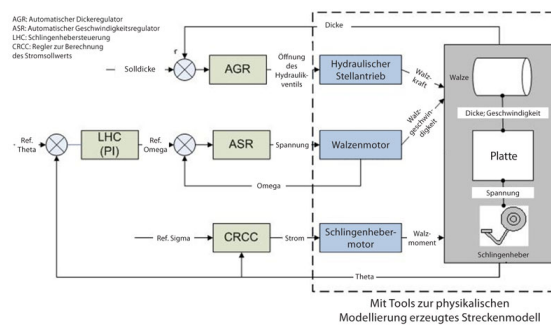


Abb. 2: Architektur des Mehrschleifenreglers. Θ = Schlingenheberwinkel, Σ = Blechspannung, Ω = Blechgeschwindigkeit.

Das Regelungssystem besteht aus folgenden Reglern:

AGR – regelt die Banddicke durch die Variation der Walzdruckkraft über die Ansteuerung des Hydraulikventils.

ASR – regelt die Bandgeschwindigkeit durch Änderung des Walzmoments, das durch die elektrische Spannung des Gleichstrommotors gesteuert wird.

LHC – stellt die Drehzollsollwerte der Walzen ein und hält so indirekt die gewünschte Materialspannung aufrecht.

CRCC – regelt die Spannung im Stahlband durch die Veränderung der Position des Schlingenhebers über den Motorstrom.

Dabei ist zu beachten, dass alle Regelkreise miteinander gekoppelt sind. So hat etwa der vom AGR-Kompensator gesteuerte hydraulische Aktuator nicht nur Einfluss auf die Banddicke, sondern auch auf die Bandgeschwindigkeit. LHC- und ASR-Regler beeinflussen gemeinsam die geforderte Spannung und Bandgeschwindigkeit.

Zunächst werden Regler entworfen, die nur auf eine einzelne Walze wirken. Hierzu wird das nichtlineare Modell mit Algorithmen aus Simulink Control Design[™] linearisiert. Mit den PID-Entwurfstools aus Simulink Control Design werden dann die Reglerparameter eingestellt. Der PID Tuner aus Simulink Control Design[™] (Abbildung 3) berechnet nach Vorgabe der gewünschten Reaktionszeit automatisch die passende PID-Parametrierung. Simulink Design Optimization[™] verfeinert danach diese Einstellungen so, dass das System unempfindlicher gegen nichtlineare Einflüsse und somit robust wird. Dieser Entwurf wird schließlich in einer nichtlinearen Simulation verifiziert. Das Regelstreckenmodell dient also zwei Zwecken: (1) Automatische Linearisierung des Streckenmodells mit Simulink Control Design und Auslegung der Regler; (2) Verifikation des geregelten nichtlinearen Gesamtsystems durch Simulation mit geschlossenen Regelkreisen.

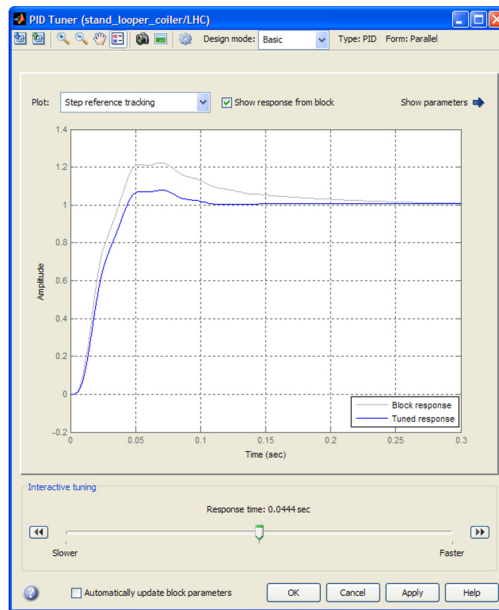


Abb. 3: Der PID Tuner. Die Abstimmung geschieht einfach per Schieberegler.

Modellierung und Simulation eines mehrstufigen Prozesses

Durch die Erzeugung von eigenen Blöcken für die einzelnen Walzstufen und des Schlingenhebers in einer Bibliothek kann man diese nun als Komponenten für den mehrstufigen Prozess wiederverwenden (Abbildung 4). Weitere Aspekte des Prozesses, wie die Massenerhaltung oder die Transportverzögerungen zwischen den Walzstufen, werden als zusätzliche Subsysteme modelliert.

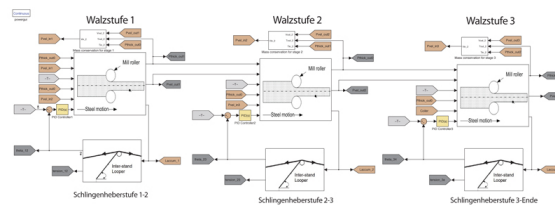


Abb. 4: Simulink-Modell aus mehreren Walzstufen mit zwischengeschalteten Schlingenhebern.

Entwurf und Verifikation einer Logik zur Fehlererkennung und -behandlung

Zusätzlich zu den Regelungssystemen müssen Prozessregler auch über Überwachungslogiken sowie Logiken zur Fehlerbehandlung verfügen, etwa um den Zustand der Sensoren und Aktuatoren im System zu überwachen. Im Folgenden wird explizit auf die Logik für die Fehlerermittlung und -behandlung eingegangen, die Fehler in hydraulischen Ventilen entdeckt und zu korrigieren versucht. Diese Logik verteilt insbesondere die Reduktion der Blechdicke auf mehrere Sollwerte für die einzelnen Walzstufen des mehrstufigen Prozesses. Sollte das Aufbringen des hydraulischen Walzendrucks an einer Stufe versagen, dann prüft die Logik, ob die anderen Stufen dies ausgleichen können. Ist dies der Fall, werden die Sollwerte der Dickenreduktion für die restlichen Stufen so berechnet, dass die gewünschte Endstärke erreicht wird.

Die Entwicklung dieser Logik findet in Stateflow[®] statt (Abbildung 5). Die hier vorgestellte Logik ist allerdings stark vereinfacht im Vergleich zu einer deutlich umfangreicheren und komplexeren Logik, die für eine echte Prozesssteuerung erforderlich wäre. In diesem Beispiel wird vereinfachend angenommen, dass die fehlerhafte Stufe im Fehlerfall gar keinen Druck mehr ausübt, das Material aber durchlaufen lässt.

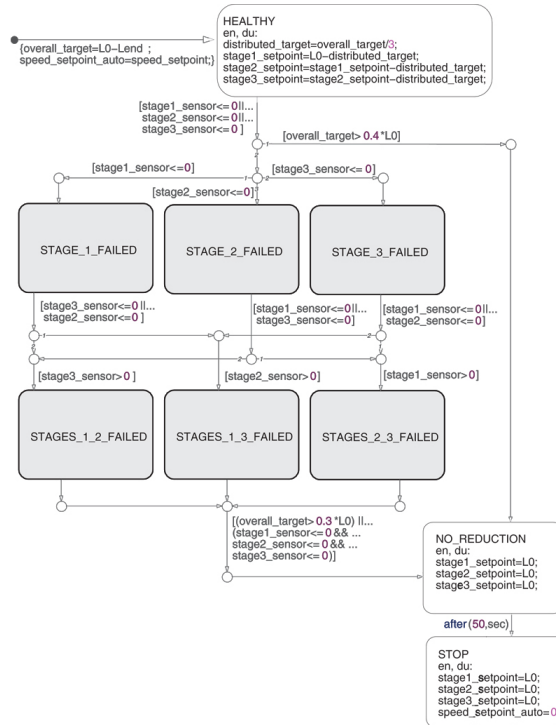


Abb. 5: Die in Stateflow implementierte fehlertolerante Sollwert-Verteilung.

Diese Logik lässt sich testen, indem man bewusst Fehler von außen in das Simulink-Modell einbringt. Abbildung 6 zeigt die Simulationsergebnisse der fehlertoleranten Logik. Versagt eine Stufe, dann überprüft der Überwachungsregler, ob die Last auf andere, funktionierende Stufen verteilt werden kann. Ist dies möglich, werden neue Sollwerte für die Dickenreduzierung an die Folgestufen-AGRs übergeben. Falls dies nicht möglich ist, wird der Prozess durch Anhalten der Bandbewegung abgeschaltet.

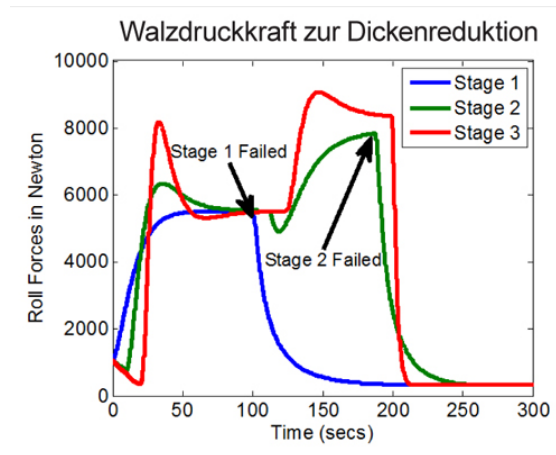


Abb. 6: Die Simulationsergebnisse zeigen, wie das System Walzstufenausfälle ausgleicht: Stufe 2 und 3 kompensieren den Ausfall von Stufe 1. Bei Ausfall von Stufe 2 wird das System abgeschaltet, da die restliche Dickenreduktion nicht alleine von Stufe 3 erreicht werden kann.

Implementierung des Reglers auf einer SPS

Mit Simulink PLC Coder™ lässt sich automatisch strukturierter Text nach IEC 61131 aus den Regler-Modellen generieren. Dieser Programmcode kann anschließend in die integrierte Entwicklungsumgebung importiert werden, die für die jeweilige SPS eingesetzt wird. Abbildung 7 zeigt den erzeugten Quellcode in strukturierter Text, der aus der Logik zur Fehlererkennung und Prozessanpassung

generiert wurde. Wie man sieht, ist der generierte Quellcode ausführlich kommentiert und lässt sich auf einfache Weise bis zu den jeweiligen Modellblöcken zurückverfolgen.

```

END_IF;
Setpoint_IN_STAGE_1_FAILED:
(* During 'STAGE_1_FAILED': '<S1>:119' *)
IF (stage3_sensor <= 0) OR (stage2_sensor <= 0) THEN
(* Transition: '<S1>:150' *)
(* Transition: '<S1>:152' *)
IF stage2_sensor > 0 THEN
(* Transition: '<S1>:155' *)
is_c2_Setpoint := Setpoint_IN_STAGES_1_3_FAILED;
(* Entry 'STAGES_1_3_FAILED': '<S1>:120' *)
rtb_stage1_setpoint := L0;
rtb_stage2_setpoint := L0 - overall_target;
distributed_target := rtb_stage2_setpoint;
ELSE
(* Transition: '<S1>:154' *)
IF stage3_sensor > 0 THEN
(* Transition: '<S1>:159' *)
is_c2_Setpoint := Setpoint_IN_STAGES_1_2_FAILED;
(* Entry 'STAGES_1_2_FAILED': '<S1>:121' *)
rtb_stage1_setpoint := L0;
rtb_stage2_setpoint := L0;
distributed_target := L0 - overall_target;
ELSE
guard_0 := TRUE;
END_IF;
END_IF;
ELSE
guard_0 := TRUE;
END_IF;

```

Abb. 7: Mit Simulink PLC Coder generierter, IEC 61131-konformer strukturierter Text. Kommentare mit direkten Referenzen zum Stateflow-Diagramm sind blau hervorgehoben.

Die automatische Codegenerierung vermeidet Fehler, die durch die manuelle Programmierung eingeschleppt werden könnten. Die numerischen Ergebnisse des auf der SPS laufenden fertigen strukturierten Texts entsprechen denen aus der Simulation. Der Simulink PLC Coder erzeugt dazu eine Testbench, mittels derer die Ergebnisse der Testausführung direkt mit denen der ursprünglichen Simulation verglichen werden können.

Möchte man die vollständig implementierte SPS-Hardware und -Software zusätzlich in einer Hardware-in-the-Loop-Simulation testen, lässt sich mit Simulink Coder™ aus dem Regelstreckenmodell auch C-Code generieren. Dieser kann auf einer Echtzeit-Simulationsumgebung wie xPC Target™ ausgeführt werden, die mit dem Regler auf der SPS kommuniziert.

Eingesetzte Produkte

- Simulink®
- SimHydraulics®
- SimMechanics™
- Simscape™
- Simulink Control Design™
- Simulink Design Optimization™
- Simulink PLC Coder™
- Stateflow®

Weitere Informationen

- Video: [Einführung in Simulink PLC Coder](#)
- [AVL entwickelt einen dynamischen Controller für ein Motorkühlsystem durch Generierung von Embedded Code für eine SPS](#)

Weitere Artikel und Newsletter-Abonnement unter www.mathworks.de/newsletters.