

# PID-Entwurf leicht gemacht

Von Murad Abu-Khalaf, Rong Chen und Arkadiy Turevskiy

DIE ABSTIMMUNG EINES PID-REGLERS SCHEINT EINFACH, MUSS MAN DOCH lediglich drei Werte finden: Die Proportional-, die Integral- und die Differential-Verstärkung. Tatsächlich aber ist die sichere und systematische Ermittlung derjenigen Verstärkungsfaktoren, mit denen ein Regelungssystem optimal arbeitet, eine sehr komplexe Aufgabe. PID-Regler werden traditionell entweder per Hand oder mit regelbasierten Methoden abgestimmt. Die manuelle Abstimmung ist zeitaufwändig und kann, wenn sie direkt an Hardware eingesetzt wird, Schäden verursachen. Regelbasierte Methoden eignen sich nicht für Regelstrecken, die Instabilitäten enthalten oder ohne Zeitverzögerung arbeiten. Die PID-Regelung birgt außerdem besondere Entwurfs- und Implementierungsprobleme wie etwa die Umsetzung zeitdiskreten Verhaltens oder eine geeignete Festkommaskalierung.

**D**ieser Artikel beschreibt am Beispiel eines Gelenkvierecks eine Methode, die den Entwurf und die Implementierung von PID-Reglern vereinfacht und verbessert. Diese Methode basiert auf den in Simulink® vorhandenen PID Controller-Blöcken und dem PID Tuning-Algorithmus aus Simulink Control Design™.

## Das Gelenkviereck: Ziele des Reglerentwurfs

Gelenkvierecke (Abb. 1) kommen in vielen mechanischen Systemen vor. Dazu gehören etwa Radaufhängungen, Aktuatoren für Roboter oder Flugzeugfahrwerke.

Das Regelungssystem besteht aus zwei Elementen: Einer Vorsteuerung (Feedforward) und einer PID-Regelung (Feedback). Die Vorsteuerung invertiert die Regelstreckendynamik – sie steuert den Haupt-

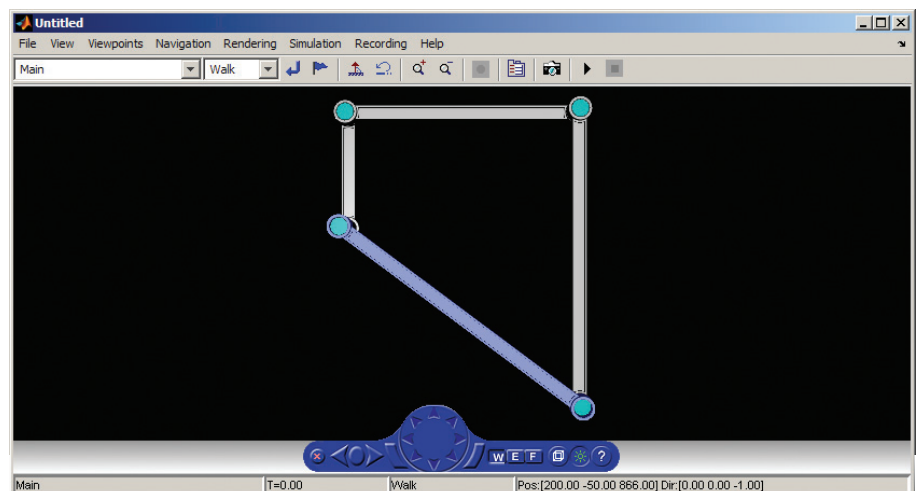


ABB. 1. Das Gelenkviereck; mit stationärem unteren Glied – dargestellt in blau.

teil der Bewegung des Mechanismus durch Berücksichtigung seines nichtlinearen Verhaltens. Die Feedback-PID-Regelung dage-

gen verringert den Positionierungsfehler, der etwa durch Modellierungsunsicherheiten und externe Störungen entsteht. Dieser

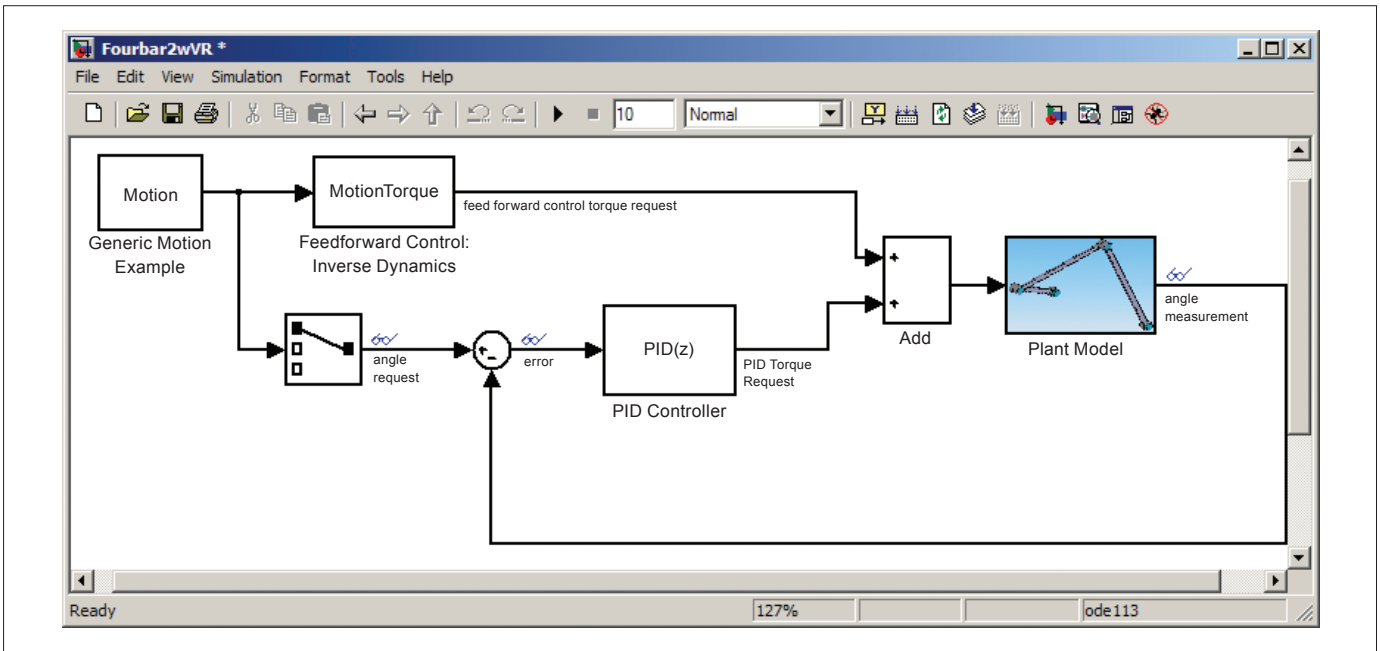


ABB. 2. Reglerarchitektur der Gelenkviereck-Steuerung.

Artikel konzentriert sich auf den Entwurf der Feedback-PID-Regelung.

Der PID-Regler bildet die Regelabweichung in Form der Differenz zwischen dem gewünschten und dem tatsächlichen Drehwinkel eines der Gelenke und fordert auf deren Grundlage ein Drehmoment an (Abb. 2). Diese Anforderung wird zur Drehmomentanforderung der Feedforward-Vorsteuerung addiert und mit diesem Summensignal wird schließlich der Gleichstrommotor angesteuert, der das Gelenk bewegt. Der Regler muss den Betrieb der Regelstrecke stabilisieren, schnell ansprechen und darf nur wenig überschwingen. Da die Implementierung auf einem Festkomputersystem mit 16-Bit erfolgen soll, muss er weiterhin zeitdiskret formuliert und seine Verstärkungsfaktoren und berechneten Signale müssen passend skaliert sein.

### Konfiguration der geschlossenen Regelschleife und Abstimmung des Reglers

Das Modell der Regelstrecke besteht aus einem in SimMechanics™ modellierten Gelenkviereck und einem in SimElectronics® mo-

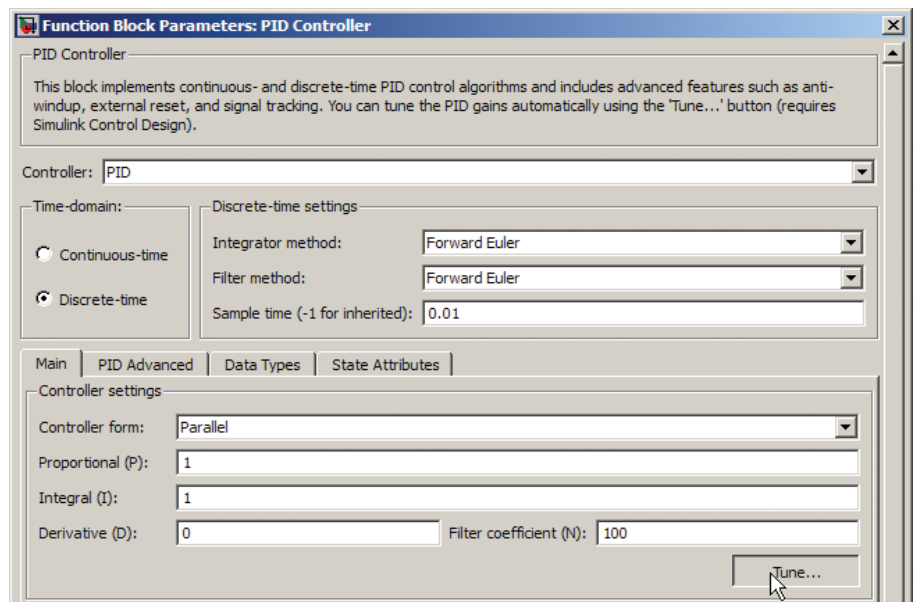


ABB. 3. Der PID Tuner, aufgerufen direkt aus der Dialogbox des Blocks.

dellierten Gleichstrommotor. Zur Erzeugung der in Abbildung 2 gezeigten Reglerarchitektur wird einfach ein zeitdiskreter PID Controller-Block aus der Simulink Discrete-Bibliothek eingefügt. Mit dieser fertigen Regelschleife kann die Abstimmung des Reglers bereits beginnen.

Man öffnet dazu die Dialogbox des PID Controller-Blocks, stellt die Abtastzeit ein und klickt auf „Tune“, worauf sich das PID Tuner öffnet (Abb. 3). Simulink Control Design linearisiert die Regelstrecke am gegenwärtigen Arbeitspunkt und leitet das lineare zeitinvariante (LTI)

Regelstreckenmodell ab, das dem PID Controller-Block in dieser Rückkopplungs-Regelschleife präsentiert wird. Die durch die Abtastung entstehende Latenz wird dabei automatisch berücksichtigt. Simulink Control Design erzeugt dann mithilfe einer automatischen Optimierungsmethode die Anfangsverstärkungen des PID-Reglers. Diese Optimierungsmethode ist unabhängig von der Ordnung der Regelstrecke oder Zeitverzögerung und funktioniert sowohl zeitkontinuierlich als auch zeitdiskret.

Abbildung 4 zeigt die Sollwertnachführung der geschlossenen Regelschleife mit diesem PID-Rohentwurf. Ist man mit der Leistung des Reglers zufrieden, werden durch einen Klick auf „Apply“ die P-, I-, D- und N-Verstärkungsfaktoren in der Dialogbox des PID Controller-Blocks aktualisiert. Das Verhalten des Reglerentwurfs lässt sich nun durch Simulation des nicht-linearen Modells und eine Visualisierung der Ergebnisse testen (Abb. 5). Man kann den Reglerentwurf aber auch interaktiv abstimmen. Dazu steht ein Schieberegler zur Verfügung, mit dem man sein Ansprechverhalten verlangsamen oder beschleunigen kann (Abb. 4).

### Vorbereitung der Implementierung

Zur Vorbereitung des Reglers auf seine Implementierung auf einem 16-Bit Mikroprozessor wird der Reglerentwurf auf die vom Prozessor unterstützte Festkomma-Arithmetik umgesetzt.

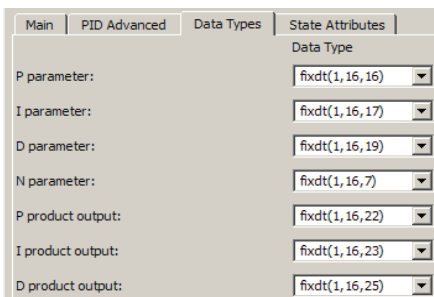


Abb. 6. Festkomma-Einstellungen für die Implementierung des PID-Reglers auf einem 16-Bit Festkomma-Prozessor.

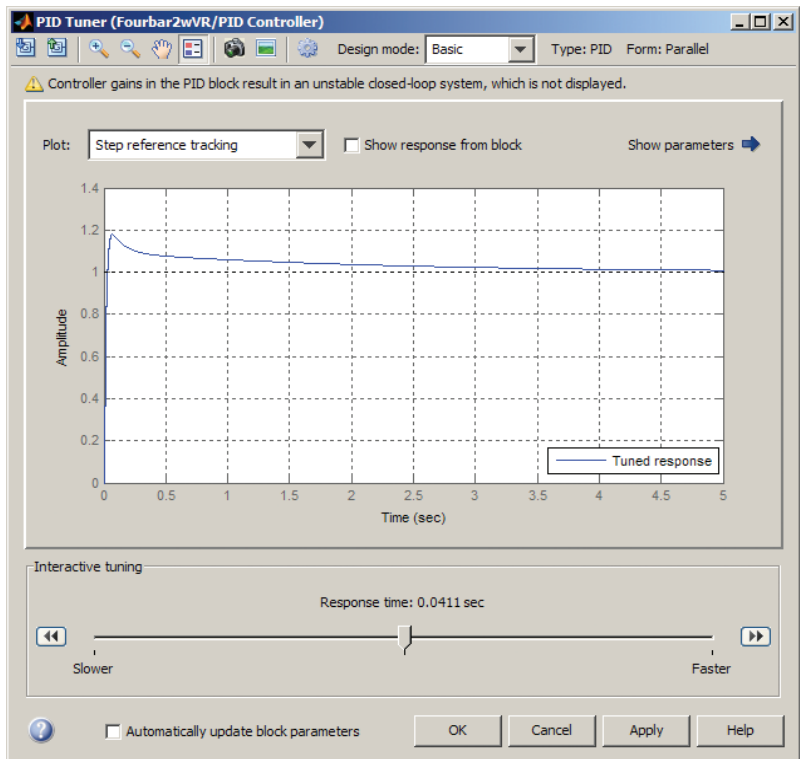


Abb. 4. Vom PID Tuner erzeugter Rohentwurf.

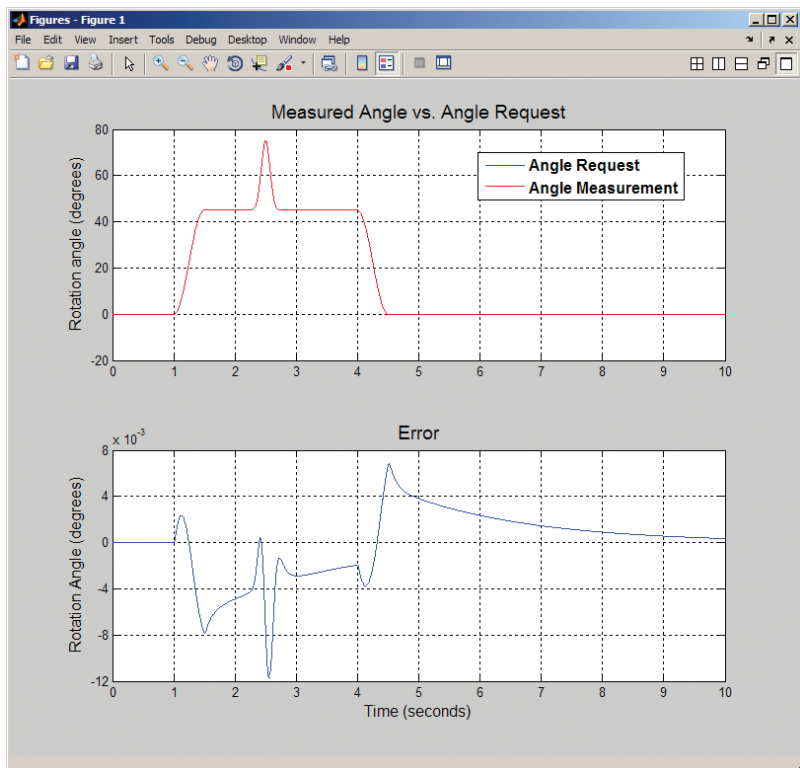


Abb. 5. Simulationsergebnisse für das Modell des Gelenkvierecks.

```

1  #include "fixed.h"
2  #include "fixed_private.h"
3
4  int16_T error;
5  int16_T torque_request;
6  D_Work DWork;
7  void fixed_step(void)
8  {
9      int16_T FilterCoefficient_m;
10     FilterCoefficient_m = (int16_T)((int32_T)((int16_T)(5403L * (int32_T)error >>
11     13U) - DWork.Filter_DSTATE) << 4U * 17893L >> 14);
12     torque_request = ((int16_T)(12475L * (int32_T)error >> 14U) >> 1) +
13     (DWork.Integrator_DSTATE >> 2)) + (FilterCoefficient_m >> 1);
14     DWork.Integrator_DSTATE = (int16_T)((4643L * (int32_T)error >> 13U) * 5243L >>
15     19U) + DWork.Integrator_DSTATE;
16     DWork.Filter_DSTATE = (int16_T)(5243L * (int32_T)FilterCoefficient_m >> 16U) +
17     DWork.Filter_DSTATE;
18 }
19
20 void fixed_initialize(void)
21 {
22     torque_request = 0;
23     (void) memset((void *)&DWork, 0,
24     sizeof(D_Work));
25     error = 0;
26 }
27

```

ABB. 7. C-Code Implementierung des 16-Bit Festkomma-Reglers. Der Code wurde direkt aus dem PID Controller-Block generiert.

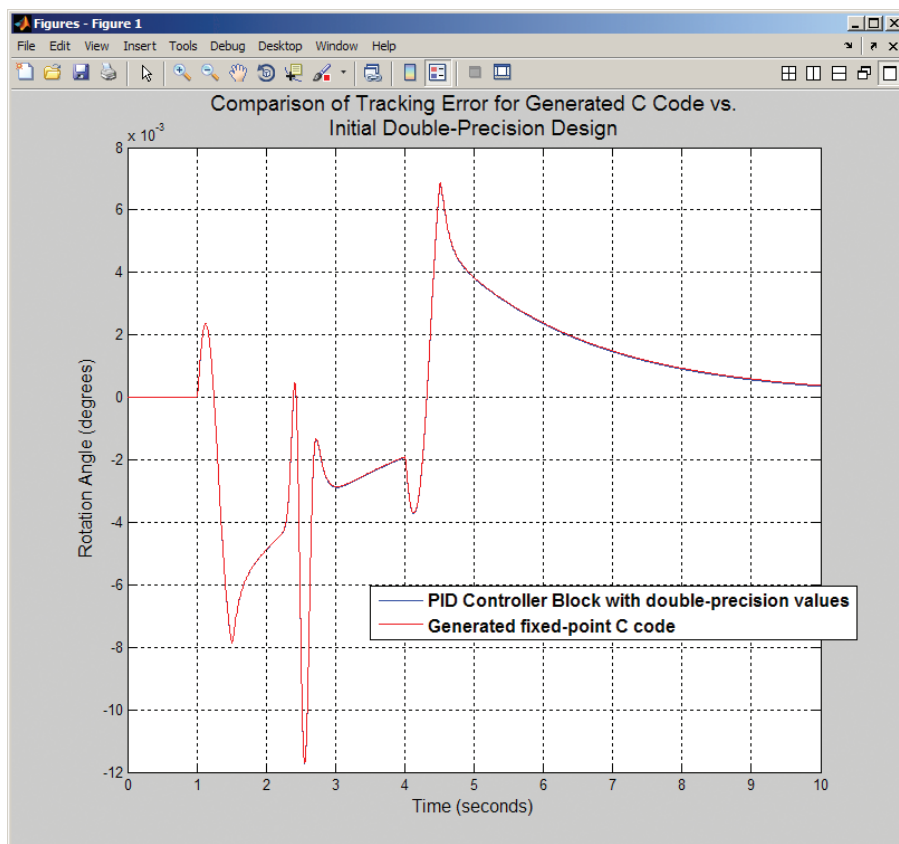


ABB. 8. Vergleich der Simulationsergebnisse des generierten C-Codes mit denen des PID Controller-Blocks mit Double-Werten.

Über den Reiter "Data Types" in der Dialogbox des Blocks werden die für den Festkommaentwurf erforderlichen Einstellungen festgelegt (Abb. 6). Dies geschieht beispielsweise automatisch mit dem Fixed-Point Tool in Simulink. Die Simulation wird nun mit den Festkommaeinstellungen wiederholt, um zu verifizieren, dass sich der Festkommaentwurf möglichst genau so verhält wie das ursprüngliche Modell, in dem die Verstärkungen und Signale durch Fließkommawerte mit doppelter Genauigkeit repräsentiert wurden.

### Generierung von Produktionscode

Der PID-Regler ist jetzt bereit für die Implementierung. Als abschließender Schritt wird mit Real-Time Workshop Embedded Coder™ C-Code generiert (Abb. 7). Zum Testen dieses Codes ersetzt man den PID Controller-Block durch den generierten C-Code und führt diesen in der geschlossenen Regelschleife aus. Der Real-Time Workshop Embedded Coder kann hierzu automatisch einen Simulink-Block erzeugen, der den generierten Code aufruft.

Die Simulation kann nun mit genau dem C-Code ausgeführt werden, der später auf dem echten Prozessor laufen wird. Abbildung 8 zeigt, dass die Ergebnisse des generierten Codes nur minimal von denen des mit Double-Werten konfigurierten PID Controller-Blocks abweichen. Der Code kann also nun auf den Prozessor heruntergeladen und zur Echtzeit-Steuerung des Gelenkvierecks eingesetzt werden. ■

### Weitere Informationen online

**DEMO: Automated Tuning of Simulink PID Controller Block**  
[www.mathworks.de/pid-tuner](http://www.mathworks.de/pid-tuner)

**DEMO: Design a Simulink PID Controller (2DOF) Block for a Reactor**  
[www.mathworks.de/pid-controller](http://www.mathworks.de/pid-controller)