



BACHELORSTUDIENGANG

Automatisierungstechnik

Simulation eines Hubwerksgtriebe

Als PROJEKTBERICHT eingereicht

zur Erlangung des akademischen Grades

Bachelor of Science in Engineering (BSc)

von

Peböck Thomas - Weindl Daniel

Wels, Jänner 2026

Betreuung der Arbeit durch

Dr. Georg Hackenberg

Inhaltsverzeichnis

1 Einleitung	1
1.1 Aufgabenstellung	1
1.2 Aufgabenverteilung	1
1.3 Vorgehensweise	1
2 Modellbildung	3
2.1 Berechnung der Bewegungsgleichung	3
2.2 Blockschaltbild	4
3 Visualisierung	5
3.1 Erstellung der Rollen	5
3.2 Erstellung von Seil und Punktmaße	5
3.3 Ergebnis	6
4 Implementierung der Regelung	7
4.1 Umsetzung in Csharp	7
4.2 Extraktion von phi	9
5 Simulation des Gesamtsystems	12
5.1 CSV laden	12
6 Fazit	13

1 Einleitung

1.1 Aufgabenstellung

Das Hubwerksgtriebe aus dem Skript MMB4 von Dr. Witteveen soll mit einem Csharp Programm simuliert werden. Dazu sind die einzelnen Komponenten des Getriebes zu modellieren und in einem Gesamtmodell zu verknüpfen. Die Simulation soll es ermöglichen, verschiedene Lastfälle durchzuspielen und die Auswirkungen auf die einzelnen Komponenten zu beobachten.

1.2.2. Beispiel: Hubwerksgtriebe mit masselosem Seil

Für das in Abbildung 7 dargestellte Hubwerk soll die die Bewegungsgleichung ermittelt werden. Alle Parameter sind gegeben. Das Schwerfeld wirkt in negative y Richtung und auf die Welle mit Index 2 wirkt das Antriebsmoment M_2 .

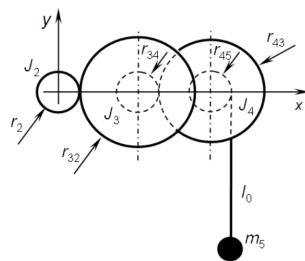


Abbildung 7: Hubwerksgtriebe

Abbildung 1.1: Aufgabe: Hubwerksgtriebe

1.2 Aufgabenverteilung

- Visualisierung des Hubwerkgetriebes - DW
- Modellierung der Bewegungsgleichung - TP
- Implementierung der Regelung - TP und DW
- Simulation des Gesamtsystems - TP und DW

1.3 Vorgehensweise

Die Idee besteht darin, mithilfe einer SFunctionContinuous eine Regelung zu simulieren. Die dabei berechneten Werte der Größe phi sollen anschließend in eine CSV-

1 Einleitung

Datei geschrieben werden, welche in weiterer Folge zur Visualisierung in OpenGL verwendet wird.

2 Modellbildung

2.1 Berechnung der Bewegungsgleichung

Die kinetische Energie des Hubwerksgtriebes schreibt sich als

$$T = \frac{1}{2}J_2\dot{\varphi}_2^2 + \frac{1}{2}J_3\dot{\varphi}_3^2 + \frac{1}{2}J_4\dot{\varphi}_4^2 + \frac{1}{2}m_5\dot{y}_5^2$$

Aus der Forderung, dass die überstrichenen Bogenlängen beim Abwälzvorgang für beide involvierten Körper gleich sind, ergibt sich

$$\varphi_2 r_2 = -\varphi_3 r_{32} \Rightarrow \varphi_3 = -\varphi_2 \frac{r_2}{r_{32}}$$

$$\varphi_3 r_{34} = -\varphi_4 r_{43} \Rightarrow \varphi_4 = \varphi_2 \frac{r_2}{r_{32}} \frac{r_{34}}{r_{43}}$$

$$y_{55} = \varphi_4 r_{45} - l_0 \Rightarrow y_{55} = \varphi_2 \frac{r_2}{r_{32}} \frac{r_{34}}{r_{43}} r_{45} - l_0$$

Das Einsetzen des Quadrates der zeitlichen Ableitungen in die kinetische Energie liefert das reduzierte Massenträgheitsmoment J_{red} .

$$T = \frac{1}{2}\dot{\varphi}_2^2 \left(J_2 + J_3 \left(\frac{r_2}{r_{32}} \right)^2 + J_4 \left(\frac{r_2 r_{34}}{r_{32} r_{43}} \right)^2 + m_5 \left(\frac{r_2 r_{34} r_{45}}{r_{32} r_{43}} \right)^2 \right) = \frac{1}{2}\dot{\varphi}_2^2 J_{\text{red}}$$

Die potentielle Energie ergibt sich zu

$$V = m_5 g \left(\varphi_2 \frac{r_2 r_{34} r_{45}}{r_{32} r_{43}} \right)$$

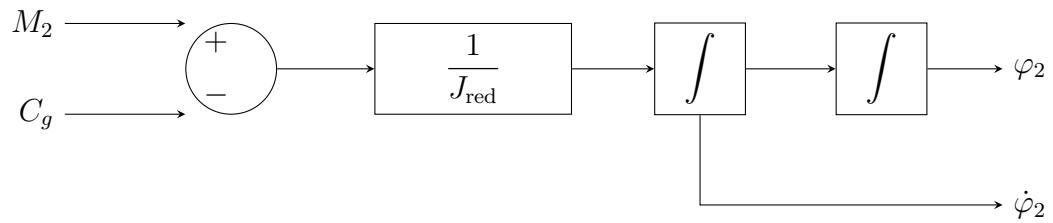
und unter Anwendung der Lagrange-Gleichung ergibt sich mit dem Antriebsmoment M_2 die Bewegungsgleichung

$$J_{\text{red}} \ddot{\varphi}_2 + m_5 g \frac{r_2 r_{34} r_{45}}{r_{32} r_{43}} = M_2$$

Simulink arbeitet am einfachsten mit expliziten Integrationsketten:

$$\ddot{\varphi}_2 = \frac{1}{J_{\text{red}}} \left(M_2 - m_5 g \frac{r_2 r_{34} r_{45}}{r_{32} r_{43}} \right)$$

2.2 Blockschaltbild



3 Visualisierung

Das Hubwerksgtriebe soll so wie in der Aufgabenstellung abgebildet werden.

3.1 Erstellung der Rollen

Jede Rolle wird nach folgendem Muster aufgebaut. Den größten Teil des Codes haben wir in die Hilfsmethode "DrawDiskXy" ausgelagert.

```
1     SetMaterial(gl, 0f, 0f, 0.5f);
2     gl.PushMatrix();
3     gl.Translate(_x2, 0, 0);
4     gl.Rotate(_phi2 * 180.0 / Math.PI, 0, 0, 1);
5     DrawDiskXy(gl, _r2, _thickness2, _segments, 0, 0, 0, ←
6           DiskMarkerColor.Red, DiskMarkerDirection.PositiveX);
    gl.PopMatrix();
```

3.2 Erstellung von Seil und Punktmaße

Die Erstellung der Punktmaße wurde auch in eine Hilfsmethode ausgelagert.

```
1 // ----- Seil -----
2 gl.Disable(OpenGL.GL_LIGHTING);
3 gl.Color(0, 0, 0);
4 gl.LineWidth(3.0f);
5 gl.Begin(OpenGL.GL_LINES);
6 gl.Vertex(x4+_r45, 0, ropeZ); // Austritt an der Rolle
7 gl.Vertex(x4+_r45, ropeY, ropeZ); // bewegte Masse
8 gl.End();
9 gl.Enable(OpenGL.GL_LIGHTING);
10 // ----- Punktmasse -----
11 SetMaterial(gl, 0, 0, 0);
12 DrawSphere(gl, 0.5, 30, 30, x4+_r45, ropeY - 0.4, ropeZ)←
;
```

3.3 Ergebnis

Die Markierungen auf den Rollen lassen gut nachvollziehen dass die Rotationen richtig übertragen werden.

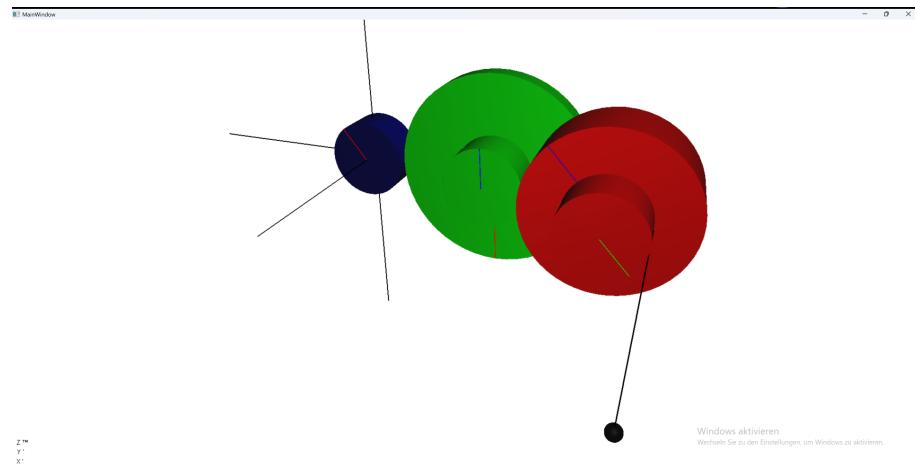


Abbildung 3.1: Ergebnis

4 Implementierung der Regelung

4.1 Umsetzung in Csharp

Die Bewegungsgleichung aus Gleichung (2.1) wird an dieser Stelle in einer SContinuousFunction umgesetzt. Hierzu wird das entsprechende Blockschaltbild nachgebildet, wie im nachfolgenden Code dargestellt.

```
1  public class Hubwerk : Example
2  {
3      public Hubwerk()
4      {
5          double _r2 = 2;
6          double _r32 = 5;
7          double _r34 = 2;
8          double _r43 = 4;
9          double _r45 = 2;
10         double m = 25;
11         double g = 9.81;
12         double jRed = 1.6;
13         double M2 = 50;
14         double a = (_r2 * _r34 * _r45) / (_r32 * _r43);
15
16         Block constantM = new ConstantBlock("Constant", ←
17             (M2/jRed));
18         Block constantC = new ConstantBlock("Constant", ←
19             ((m*g*a)/jRed));
20         Block sub = new SubtractBlock("Sub");
21         Block integrate1 = new IntegrateBlock("←
22             Integrate1", 0);
23         Block integrate2 = new IntegrateBlock("←
24             Integrate2", 0);
25         Block recordphi = new RecordBlock("Phi");
26         Block recordphidot = new RecordBlock("←
27             Winkelgeschwindigkeit");
28         Block recordphidotdot = new RecordBlock("←
29             Winkelbeschleunigung");
```

```

25     Model.AddBlock(constantM);
26     Model.AddBlock(constantC);
27     Model.AddBlock(sub);
28     Model.AddBlock(integrate1);
29     Model.AddBlock(integrate2);
30     Model.AddBlock(recordphi);
31     Model.AddBlock(recordphidot);
32     Model.AddBlock(recordphidotdot);
33
34     Model.AddConnection(constantM, 0, sub, 0);
35     Model.AddConnection(constantC, 0, sub, 1);
36     Model.AddConnection(sub, 0, integrate1, 0);
37     Model.AddConnection(integrate1, 0, integrate2, ←
38         0);
39     Model.AddConnection(integrate2, 0, recordphi, 0)←
40         ;
41     Model.AddConnection(integrate1, 0, recordphidot, ←
42         0);
        Model.AddConnection(sub, 0, recordphidotdot, 0);
    }
}

```

Wir lassen das Example Hubwerk bauen:

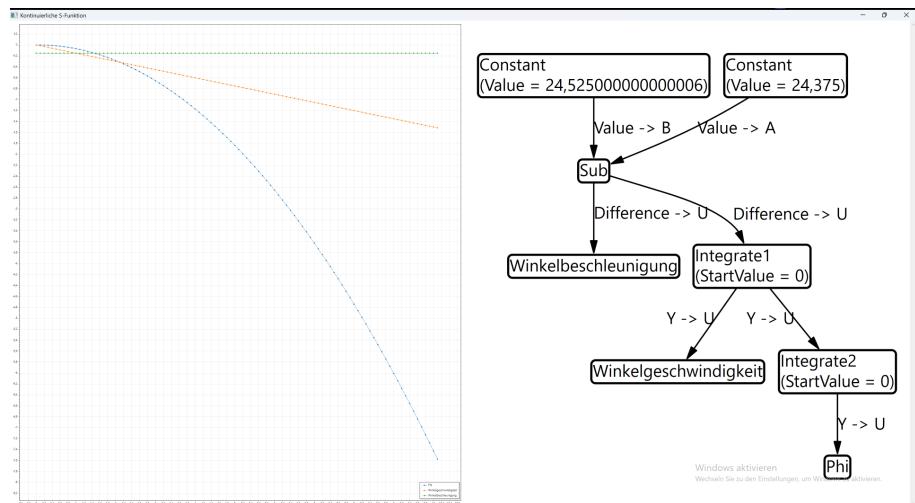


Abbildung 4.1

4.2 Extraktion von phi

Nach mehreren Tests wurde die Plausibilität der Modellbildung als bestätigt angesehen. Im nächsten Schritt soll die Liste der Variable phi in eine CSV-Datei exportiert werden.

Den Code dafür binden wir direkt nach dem Solver ein.

```
1
2 namespace SFunctionContinuous
3 {
4     /// <summary>
5     /// Interaction logic for MainWindow.xaml
6     /// </summary>
7     public partial class MainWindow : Window
8     {
9         public MainWindow()
10        {
11            InitializeComponent();
12
13            // Modell erstellen und loesen
14
15            //Example example = new PosServo();
16            Example example = new Hubwerk();
17
18            try
19            {
20                Solver solution = new EulerExplicitSolver(<-
21                    example.Model);
22                solution.Solve(example.TimeStepMax, example.<-
23                    TimeMax);
24            }
25            catch (Exception e)
26            {
27                MessageBox.Show(e.Message);
28            }
29            // =====
30            // CSV-EXPORT VON PHI (HIER!)
31            // =====
32
33            RecordBlock phiBlock = example.Model.Blocks
```

```
32         .OfType<RecordBlock>()
33         .First(b => b.Name == "Phi");
34
35     string filePath = @"Z:\_source\Repositorys\←
36           SSI_Hubwerksgetriebe\phi2.csv";
37
38     CultureInfo culture = CultureInfo.←
39           InvariantCulture;
40     String Builder sb = new String Builder();
41
42     sb.AppendLine("phi");
43
44     foreach ((_, double phi) in phiBlock.Data)
45     {
46         sb.AppendLine(phi.ToString(culture));
47     }
48
49     File.WriteAllText(filePath, sb.ToString());
50
51     // Graph-Visualisierung erstellen
52
53     Graph graph = new Graph();
54
55     foreach (Block f in example.Model.Blocks)
56     {
57         graph.AddNode($"{f.GetHashCode()}").←
58             LabelText = f.ToString();
59     }
60
61     foreach (Connection c in example.Model.←
62             Connections)
63     {
64         graph.AddEdge($"{c.Source.GetHashCode()}", c.←
65             .ToString(), $"{c.Target.GetHashCode()}")←
66             ;
67     }
68
69     Graph.Graph = graph;
70
71     // Chart-Visualisierung erstellen
72
73
74
75
```

```
66     foreach (Block f in example.Model.Blocks)
67     {
68         if (f is RecordBlock)
69         {
70             RecordBlock r = (RecordBlock)f;
71
72             double[] t = new double[r.Data.Count];
73             double[] u = new double[r.Data.Count];
74
75             for (int i = 0; i < r.Data.Count; i++)
76             {
77                 (double ti, double tu) = r.Data[i];
78
79                 t[i] = ti;
80                 u[i] = tu;
81             }
82
83             Scatter scatter = Chart.Plot.Add.Scatter←
84                         (t, u);
85
86             scatter.LegendText = f.Name;
87         }
88     }
89 }
90 }
```

Jetzt muss die Visualisierung nur noch auf die Liste zugreifen.

5 Simulation des Gesamtsystems

5.1 CSV laden

Die CSV-Datei wird direkt im Konstruktor geladen. Über den phiTimer wird in einem Intervall von 0,1 s jeweils eine neue Zeile aus der CSV eingelesen. Auf diese Weise wird das zuvor simulierte Modell an dieser Stelle visualisiert.

```
1     public MainWindow()
2     {
3         InitializeComponent();
4
5         // CSV laden
6         LoadPhi2FromCsv(@"Z:\_source\Repositorys\←
7             SSI_Hubwerksgtriebe\phi2.csv");
8
9         // Timer starten
10        _phiTimer = new DispatcherTimer();
11        _phiTimer.Interval = TimeSpan.FromSeconds(0.1);
12        _phiTimer.Tick += PhiTimer_Tick;
13        _phiTimer.Start();
14    }
```

Ein Video der Visualisierung liegt dem Bericht bei.

6 Fazit

Durch die Trennung von Berechnung und Visualisierung war es möglich, beide Entwicklungsbereiche parallel zu starten und unabhängig voneinander weiterzuentwickeln. Die einzige Verbindung zwischen diesen beiden Bereichen stellt die CSV-Datei dar.

Die Umsetzung erwies sich als anspruchsvoller als ursprünglich angenommen. Für die Visualisierung wurde zwar KI-Unterstützung genutzt, dennoch waren mehrere Anpassungen erforderlich. Ein grundlegendes Verständnis des Codes sowie eine durchdachte Architektur mussten in jedem Fall selbst erarbeitet werden.

Aus diesem Grund wurden an den Rollen Markierungen angebracht, um die Rotationsbewegungen besser nachvollziehen und überprüfen zu können.

test test