Projektarbeit

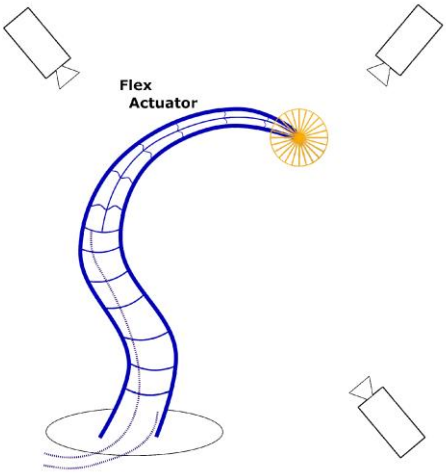
Magnetisches Tracking-System

ZHAO, Haibin

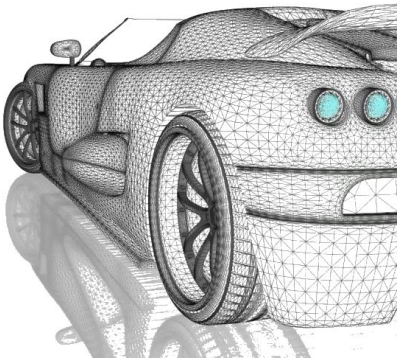
01.April.2020

1 Motivation

Magnetische Tracking-Systeme sind passive Tracking-Systeme. Sie werden in der medizinischen Robotik breit angewendet.



Im Vergleich zu optischen Tracing-Systemen können magnetische Tracking-Systeme bei unsichtbaren Situationen auch benutzen, z.B. In-vivo Mikrorobotern.



Im Vergleich zu aktiven Tracking-Systemen verbrauchen die zu loklisierenden Agenten dafür keine Energie. Demzufolge können die Roboter klein, leicht und mobil zu bleiben.

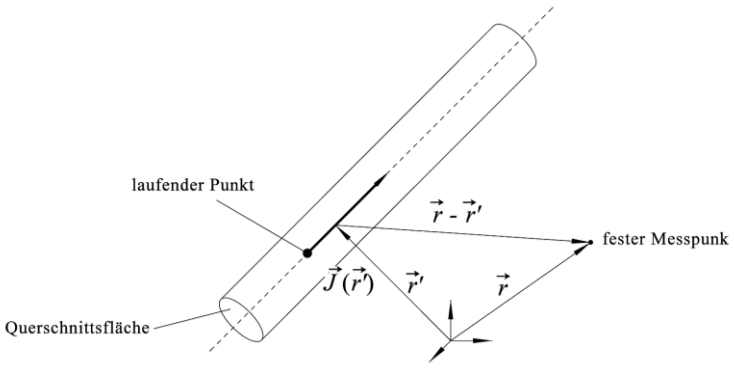
2 Modellierung von zylindrischen Magneten

2.1 Analytische Formel

2.1.1 Grundlagen

Mit Hilfe magnetischen Dipol-Modells

Laut Biot-Savart-Gesetz



$$\vec{B}(\vec{r}) = \frac{\mu}{4\pi} \iiint_V \frac{\vec{J}(\vec{r}') \times (\vec{r} - \vec{r}')}{|\vec{r} - \vec{r}'|^3} \mathrm{d}V$$

wobei

$\vec{B}$ Magnetische Flussdichte

$\vec{J}$ Stromdichte

Vereinfach nun $\vec{r}'$ zu Null, also der Leiter liegt genau am Koordinaten-Ursprung, dann folgt:

$$\vec{B}(\vec{r}) = \frac{\mu}{4\pi} \iiint_V \frac{\vec{J}(\vec{0}) \times \vec{r}}{|\vec{r}|^3} \mathrm{d}V$$

Magnetische Flussdichte $\vec{B}$ besteht aus 2 Teilen, also

$$\vec{B} = \overrightarrow{B_I} + \overrightarrow{B_M}$$

$\vec{B}_I$ ist die durch freie Ströme I erzeugten Felde

$\overrightarrow{B_M}$ ist die durch gebundene atomare Kreisströme erzeugten Magnetisierung

Mithilfe der Definition $\vec{H} = \vec{B}_I/\mu_0$ und der Beziehung $\vec{B} = \mu_r \vec{B}_I$ ergibt sich

$$\vec{B} = \mu_r \vec{B}_I = \underbrace{\mu_0 \mu_r}_{=:\mu} \vec{H} =: \mu \vec{H}$$

wobei

μ_0 magnetische Feldkonstante

μ_r den Anteil von B_M am Gesamtfeld beinhaltet

Zerlegen nun die Stromdichte $\vec{J}(\vec{0})$ in 2 Teilen, nämlich Volumenstromdichte und Oberflächenstromdichte. Dann erhält man das magnetische Feld von zylindrischem Permanentmagneten

$$\vec{H} = \frac{1}{4\pi} \left(\iiint_V \frac{\rho_m \vec{R}}{R^3} \mathrm{d}V + \oiint_S \frac{\rho_{ms} \vec{R}}{R^3} \mathrm{d}S \right)$$

mit $\vec{H}$ Magnetfeldstärke

$\rho_m = -\operatorname{div} \vec{M}$ Volumenstromdichte

$\operatorname{div} \vec{M} = \frac{\partial M_x}{\partial x} + \frac{\partial M_y}{\partial y} + \frac{\partial M_z}{\partial z}$ Divergenz

$\rho_{ms} = \overrightarrow{e_n} \cdot \vec{M}$ Flächenstromdichte

$\vec{M}$ Magnetisierung. Für eine spezifische Magnet ist ein Konstant $\vec{M} = [m_1, m_2, m_3]_{\vec{i}}$

$\overrightarrow{e_n}$ Oberfläche-Normvektor

$\vec{R}$ Vektor von Magneteinheit nach zurechneten Punkt

R Abstand von Magneteinheit nach zurechneten Punkt

V zu integrierte Volumen von Dauermagnet

S zu integrierte Oberfläche von Dauermagnet

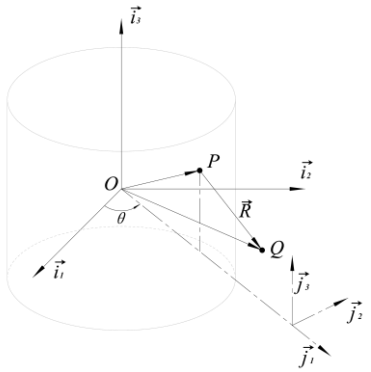
2.1.2 Koordinatensystem

Bilden wir nun 2 Koordinaten Systeme:

I-System $(\vec{\imath_1}, \vec{\imath_2}, \vec{\imath_3})$ für Magnet-Koordinaten

J-System $(\vec{J_1}, \vec{J_2}, \vec{J_3})$ für zu berechneten Punkten

Außerdem wird ein zylindrisches Koordinatensystem (θ, r, h) gebildet (nicht gezeigt in folgendem Bild).



hier $P = [x, y, z]_{\vec{\imath}}$ ist eine Magneteinheit, $Q = [q_1, q_2, q_3]_{\vec{\imath}}$ ist ein zu berechneter Punkt

Koordinaten Transformation

Die Transformationsmatrix zwischen $\underline{\vec{\imath}}$ und $\underline{\vec{j}}$:

$$\underbrace{\underline{\vec{\imath}} = \begin{bmatrix} \vec{\imath_1} \\ \vec{\imath_2} \\ \vec{\imath_3} \end{bmatrix}}_{=:\underline{\vec{\imath}}} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{=:{}^{I}\underline{m}^{J}} \cdot \underbrace{\begin{bmatrix} \vec{J_1} \\ \vec{J_2} \\ \vec{J_3} \end{bmatrix}}_{=:\underline{\vec{j}}} =: {}^{I}\underline{m}^{J} \cdot \underline{\vec{j}}$$

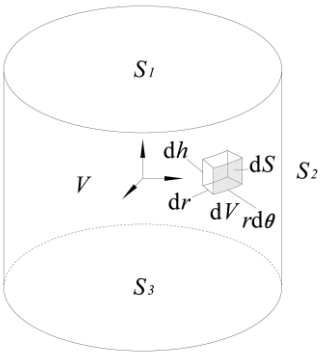
dann folgt offenbar

$$\begin{aligned} \vec{R} &= \overrightarrow{OQ} - \overrightarrow{OP} \\ &= [q_1, q_2, q_3] \cdot \underline{\vec{\imath}} - [x, y, z] \cdot \underline{\vec{\imath}} \\ &= \begin{bmatrix} q_1 - x \\ q_2 - y \\ q_3 - z \end{bmatrix}^{\mathrm{T}} \cdot \underline{\vec{\imath}} \end{aligned}$$

und natürlich

$$R^3 = [(q_1 - x)^2 + (q_2 - y)^2 + (q_3 - z)^2]^{\frac{3}{2}}$$

Herleitung der Integrationsformel



mit der Beziehung $x = r\cos\theta,\ y = r\sin\theta, h = z$, gilt

$$\begin{aligned}
&\iiint_V \frac{\rho_m \vec{R}}{R^3}\mathrm{d}V \\
&= \iiint_V \frac{\rho_m[(q_1-x)\vec{\imath_1} + (q_2-y)\vec{\imath_2} + (q_3-z)\vec{\imath_3}]}{[(q_1-x)^2+(q_2-y)^2+(q_3-z)^2]^{\frac{3}{2}}}\mathrm{d}V \\
&= \begin{bmatrix} \iiint_V \dfrac{\rho_m(q_1 - r\cos\theta)}{[(q_1-r\cos\theta)^2+(q_2-r\sin\theta)^2+(q_3-h)^2]^{\frac{3}{2}}} r\mathrm{d}r\mathrm{d}\theta\mathrm{d}h \\ \iiint_V \dfrac{\rho_m(q_2 - r\sin\theta)}{[(q_1-r\cos\theta)^2+(q_2-r\sin\theta)^2+(q_3-h)^2]^{\frac{3}{2}}} r\mathrm{d}r\mathrm{d}\theta\mathrm{d}h \\ \iiint_V \dfrac{\rho_m(q_3 - h)}{[(q_1-r\cos\theta)^2+(q_2-r\sin\theta)^2+(q_3-h)^2]^{\frac{3}{2}}} r\mathrm{d}r\mathrm{d}\theta\mathrm{d}h \end{bmatrix}^{\mathrm{T}}_{\vec{\imath}}
\end{aligned}$$

Für Dauermagneten ist wegen der Homogenität der Teil von Volumenintegration gleich Null. Beweis dafür ist:

$$\begin{aligned}
\rho_m &= -\mu_0 \,\mathrm{div}\,\vec{M} \\
&= -\mu_0\left(\frac{\partial M_x}{\partial x} + \frac{\partial M_y}{\partial y} + \frac{\partial M_z}{\partial z}\right) \\
&= -\mu_0\left(\frac{\partial m_1}{\partial x} + \frac{\partial m_2}{\partial y} + \frac{\partial m_3}{\partial z}\right) \\
&= 0
\end{aligned}$$

Deswegen gilt

$$\iiint_V \frac{\rho_m \vec{R}}{R^3}\mathrm{d}V = 0$$

Für den zweiten Teil:

Für S_1 gilt

$$\begin{aligned}
\rho_{ms}\vec{R} &= \left(\overrightarrow{e_n}\cdot\vec{M}\right)\cdot\vec{R} \\
&= \left[\left(\begin{bmatrix}0\\0\\1\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\imath}}\right)^{\mathrm{T}}\cdot\begin{bmatrix}m_1\\m_2\\m_3\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\imath}}\right]^{\mathrm{T}}\cdot\vec{R} \\
&= \left[\underline{\vec{\imath}}^{\mathrm{T}}\cdot\begin{bmatrix}0\\0\\1\end{bmatrix}\cdot\begin{bmatrix}m_1\\m_2\\m_3\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\imath}}\right]^{\mathrm{T}}\cdot\vec{R} \\
&= [\overrightarrow{\imath_1}\quad\overrightarrow{\imath_2}\quad\overrightarrow{\imath_3}]\begin{bmatrix}m_1\\m_2\\m_3\end{bmatrix}[0\quad 0\quad 1]\begin{bmatrix}\overrightarrow{\imath_1}\\\overrightarrow{\imath_2}\\\overrightarrow{\imath_3}\end{bmatrix}\cdot\left(\begin{bmatrix}q_1-x\\q_2-y\\q_3-z\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\imath}}\right) \\
&= m_3\cdot\begin{bmatrix}q_1-x\\q_2-y\\q_3-z\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\imath}}
\end{aligned}$$

Hier $z = h_{S_1} = L/2$, wobei L die gesamte Hohe des Dauermagnets.

Für S_3 gilt analog

$$\rho_{ms}\vec{R} = -m_3\cdot\begin{bmatrix}q_1-x\\q_2-y\\q_3-z\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\imath}}$$

Hier $z = h_{S_3} = -L/2$.

Für S_2 gilt

$$
\begin{aligned}
\rho_{ms}\vec{R} &= \left(\overrightarrow{e_n}\cdot\vec{M}\right)\cdot\vec{R} \\
&= \left[\left(\begin{bmatrix}1\\0\\0\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{J}}\right)^{\mathrm{T}}\cdot\begin{bmatrix}m_1\\m_2\\m_3\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\iota}}\right]^{\mathrm{T}}\cdot\vec{R} = \left[\underline{\vec{\iota}}^{\mathrm{T}}\cdot{}^{I}\underline{m}^{J}\begin{bmatrix}1\\0\\0\end{bmatrix}\cdot\begin{bmatrix}m_1\\m_2\\m_3\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\iota}}\right]^{\mathrm{T}}\cdot\vec{R} \\
&= [\overrightarrow{\iota_1}\quad\overrightarrow{\iota_2}\quad\overrightarrow{\iota_3}]\begin{bmatrix}m_1\\m_2\\m_3\end{bmatrix}[1\quad 0\quad 0]\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\overrightarrow{\iota_1}\\\overrightarrow{\iota_2}\\\overrightarrow{\iota_3}\end{bmatrix}\cdot\left(\begin{bmatrix}q_1-x\\q_2-y\\q_3-z\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\iota}}\right) \\
&= (m_1\cos\theta + m_2\sin\theta)\begin{bmatrix}q_1-x\\q_2-y\\q_3-z\end{bmatrix}^{\mathrm{T}}\cdot\underline{\vec{\iota}}
\end{aligned}
$$

Hier $x = R_a\cos\theta,\ y = R_a\sin\theta$, wobei R_a der Halbmesser des Dauermagnets.

Zur Vereinfachung betrachten wir wegen der Rotationssymmetrie der Magnetfeldverteilung nur die Magnetfeldstärke in $\overrightarrow{\iota_2}-\overrightarrow{\iota_3}$ Ebene, d.h. $q_1 \overset{!}{=} 0$. Dann gilt die folgende Herleitung:

$$
\begin{aligned}
&\frac{1}{4\pi}\oiint_S\frac{\rho_{ms}\vec{R}}{R^3}\mathrm{d}S \\
&= \frac{1}{4\pi}\iint_{S_1}\frac{\rho_{ms}\vec{R}}{R^3}\mathrm{d}S + \frac{1}{4\pi}\iint_{S_2}\frac{\rho_{ms}\vec{R}}{R^3}\mathrm{d}S + \frac{1}{4\pi}\iint_{S_3}\frac{\rho_{ms}\vec{R}}{R^3}\mathrm{d}S \\
&= \frac{1}{4\pi}\iint_{S_1}\frac{m_3\cdot\left[(-r\cos\theta)\overrightarrow{\iota_1}+(q_2-r\sin\theta)\overrightarrow{\iota_2}+\left(q_3-\frac{L}{2}\right)\overrightarrow{\iota_3}\right]}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3-\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta \\
&\quad+\underbrace{\frac{1}{4\pi}\iint_{S_2}\frac{(m_1\cos\theta+m_2\sin\theta)[(-R_a\cos\theta)\overrightarrow{\iota_1}+(q_2-R_a\sin\theta)\overrightarrow{\iota_2}+(q_3-h)\overrightarrow{\iota_3}]}{[(R_a\cos\theta)^2+(q_2-R_a\sin\theta)^2+(q_3-h)^2]^{\frac{3}{2}}}R_a\mathrm{d}\theta\mathrm{d}h}_{=0} \\
&\quad+\frac{1}{4\pi}\iint_{S_3}\frac{-m_3\cdot\left[(-r\cos\theta)\overrightarrow{\iota_1}+(q_2-r\sin\theta)\overrightarrow{\iota_2}+\left(q_3+\frac{L}{2}\right)\overrightarrow{\iota_3}\right]}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3+\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta \\
&= \begin{bmatrix}
\frac{m_3}{4\pi}\int_0^{2\pi}\int_0^{R_a}\frac{(-r\cos\theta)}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3-\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta\\
\frac{m_3}{4\pi}\int_0^{2\pi}\int_0^{R_a}\frac{(q_2-r\sin\theta)}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3-\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta\\
\frac{m_3}{4\pi}\int_0^{2\pi}\int_0^{R_a}\frac{\left(q_3-\frac{L}{2}\right)}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3-\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta
\end{bmatrix}_{\vec{\iota}} \\
&\quad+\begin{bmatrix}
\frac{-m_3}{4\pi}\int_0^{2\pi}\int_0^{R_a}\frac{(-r\cos\theta)}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3+\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta\\
\frac{-m_3}{4\pi}\int_0^{2\pi}\int_0^{R_a}\frac{(q_2-r\sin\theta)}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3+\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta\\
\frac{-m_3}{4\pi}\int_0^{2\pi}\int_0^{R_a}\frac{\left(q_3+\frac{L}{2}\right)}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3+\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta
\end{bmatrix}_{\vec{\iota}} \\
&=: \begin{bmatrix}H_{11}\\H_{12}\\H_{13}\end{bmatrix}_{\vec{\iota}}^{T} + \begin{bmatrix}H_{31}\\H_{32}\\H_{33}\end{bmatrix}_{\vec{\iota}}^{T}
\end{aligned}
$$

Offenbar ist die Magnetfeldstärke in $\vec{\imath_1}$ Richtung wegen der Rotationssymmetrie gleich Null. D.h.

$$\int_0^{2\pi}\int_0^{R_a}\frac{m_3\cdot(-r\cos\theta)}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3-\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta + \int_0^{R_a}\int_0^{2\pi}\frac{-m_3\cdot(-r\cos\theta)}{\left[(r\cos\theta)^2+(q_2-r\sin\theta)^2+\left(q_3+\frac{L}{2}\right)^2\right]^{\frac{3}{2}}}r\mathrm{d}r\mathrm{d}\theta = 0$$

2.2 Numerische Lösung

2.2.1 Numerische Berechnung

Annahme:

Magnet Größe $R_a = 9$ mm, $L = 5$ mm

Magnetwerkstoff: (https://www.neomagnete.de/de/magnetisierungsgrad)

Magnetisierungsgrad N52

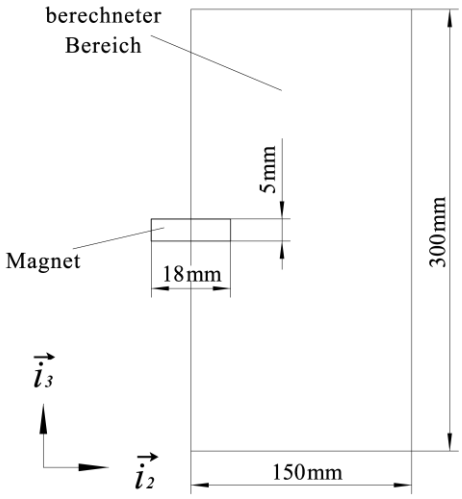
Magnetische Remanenz $B_r = 1480\ \mathrm{mT} = 1.48\ \mathrm{T}$

Berechnung von Magnetfeldstärke

Magnetisierung: Da der Permanentmagnet homogen magnetisiert wird, gilt

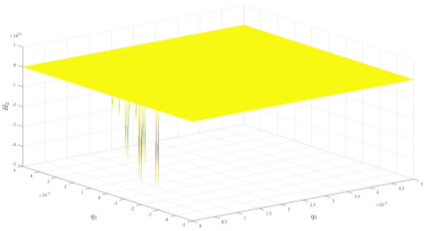
$$m_3 = \frac{B_r}{\mu_0} = \frac{1.48}{4\pi\times10^{-7}}\frac{\mathrm{T}}{\mathrm{T\cdot m/A}} \approx 1.178\times10^6\ \mathrm{A/m}$$

Berechneter Bereich: wegen der Rotationssymmetrie wird ein $150\ \mathrm{mm}\times300\ \mathrm{mm}$ berechnet.

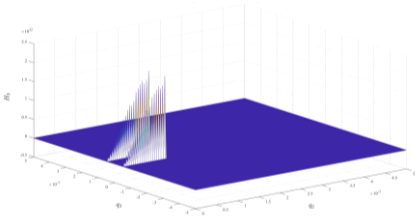


Resultat

Magnetfeldstärkeverteilung in $\vec{\imath_2}$ Richtung



Magnetfeldstärkeverteilung in $\vec{\imath_3}$ Richtung



2.2.2 Bereinigung

Die Magnetfeldstärke innerhalb des Magnets ist so groß, dass die Magnetfeldstärkeverteilung an anderen Stellen nicht offensichtlich ist. Deswegen setzen wir die Magnetfeldstärke innerhalb des Magnets zu Null, weil die Magnetfeldstärke dort kein Sinn macht.

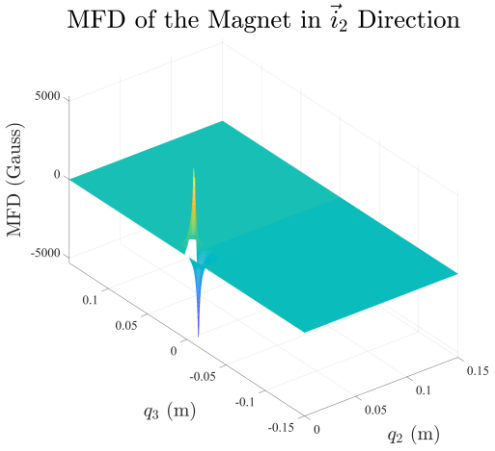
Außerdem rechnen wir MFS in Magnet Fluss Dichte um.

Für Permanentmagnete gilt die folgende Beziehung:

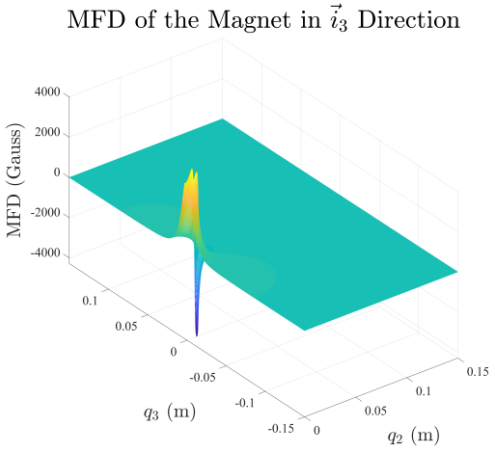
$$B = \mu_0\ (\mathrm{N/A^{-2}}) \cdot H\ (\mathrm{A/m}) = \mu_0 \cdot H\ (\mathrm{T}) = 10000 \cdot \mu_0 \cdot H \cdot (\mathrm{Gauß})$$

Nach der Bereinigung erhalten wir:

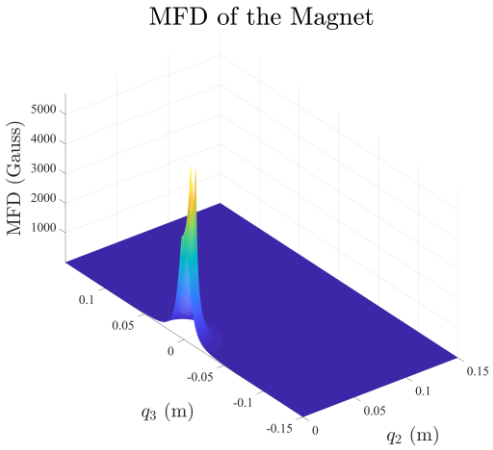
MFD in $\vec{\imath_2}$ Richtung (Verteilung von B_2)



MFD in $\vec{\imath_3}$ Richtung (Verteilung von B_3)



Absolute MFD (Verteilung von $B_m = \sqrt{B_2^2 + B_3^2}$)



2.2.3 MFD von einem langen Magnet

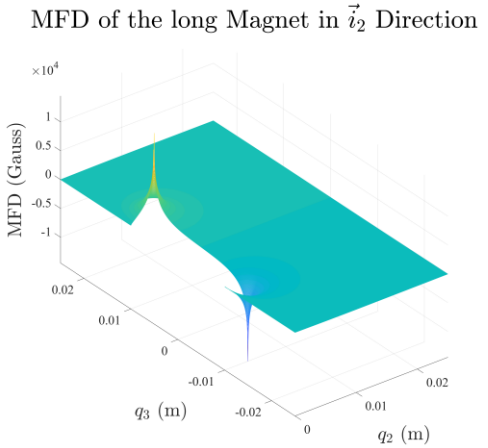
Magnet Größe $R_a = 3.75$ mm, $L = 20$ mm

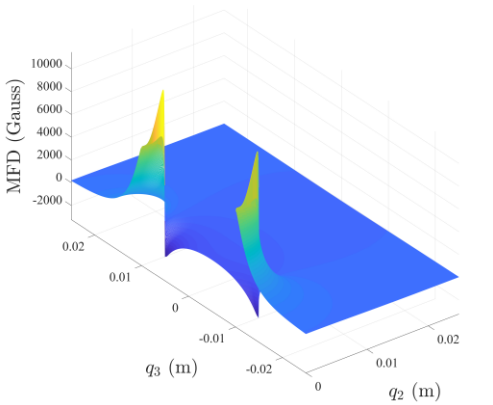
Magnetwerkstoff: (https://www.neomagnete.de/de/magnetisierungsgrad)

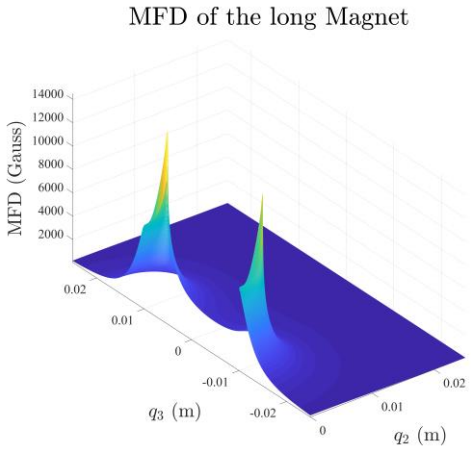
Magnetisierungsgrad N52

Magnetische Remanenz $B_r = 1480\,\text{mT} = 1.48\,\text{T}$

Ergebnisse:



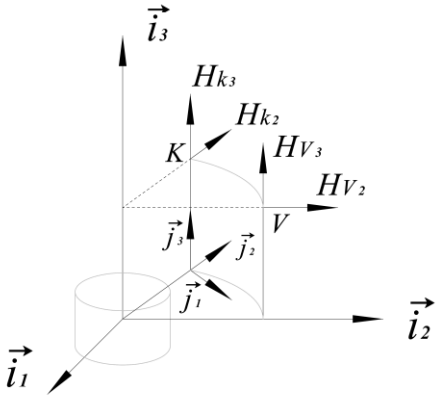




2.3 Magnetkoordinaten Transformation

Da wir nur die MFS in einer Ebene berechnet haben, ist eine zusätzliche Koordinaten-Transformation benötigt, um die MFS von jedem Punkt um das Magnet zu berechnen.

Magnetfeldstärke an beliebigen Punkt K lässt sich aus den sogenannte „Vergleichspunkt“ V berechnen.



Für einen beliebigen Punkt $K = [k_1 \quad k_2 \quad k_3]_{\vec{i}}$ gilt

$$^{J}\vec{r}^{K} = [k_1 \quad k_2 \quad k_3] \cdot \underline{\vec{i}} = [k_1 \quad k_2 \quad k_3] \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \underline{\vec{J}} = \begin{bmatrix} k_1\cos\theta + k_2\sin\theta \\ -k_1\sin\theta + k_2\cos\theta \\ k_3 \end{bmatrix}_{\underline{\vec{J}}}$$

Wir rotieren nun Koordinaten System J so, dass der Punkt K auf der Ebene $\vec{J_2} - \vec{J_3}$, auf 1. und 4. Quadrant fällt, also

$$k_1\cos\theta_k + k_2\sin\theta_k \overset{!}{=} 0$$
$$-k_1\sin\theta_k + k_2\cos\theta_k \overset{!}{\geq} 0$$

erhalten wir

$$\theta_k = -\operatorname{atan}\frac{k_1}{k_2} + \delta \cdot \pi$$

mit

$$\delta = \begin{cases} -1, \text{für } k_2 < 0, k_1 > 0 \\ 0, \quad \text{für } k_2 \geq 0 \\ 1, \quad \text{für } k_2 < 0, k_1 \leq 0 \end{cases}$$

also jetzt gilt

$$^{J}\vec{r}^{K} = \begin{bmatrix} 0 \\ -k_1\sin\theta_k + k_2\cos\theta_k \\ k_3 \end{bmatrix}_{\underline{\vec{J}}}$$

Wegen der Rotationssymmetrie ist die Magnetfeldstärke am Punkt ${}^{I}\vec{r}^{K}$ betragsmäßig gleich wie die am Vergleichspunkt V, wobei

$$ {}^{I}\vec{r}^{V} = \begin{bmatrix} 0 \\ -k_1 \sin\theta_k + k_2 \cos\theta_k \\ k_3 \end{bmatrix}_{\underline{\vec{\imath}}} =: \begin{bmatrix} 0 \\ v_2 \\ v_3 \end{bmatrix}_{\underline{\vec{\imath}}} $$

Die Magnetfeldstärke am ${}^{I}\vec{r}^{V}$ ist

$$ \vec{B}(V) = \begin{bmatrix} 0 \\ B_2(V) \\ B_3(V) \end{bmatrix}_{\underline{\vec{\imath}}} $$

Die Beziehung zwischen Vergleichspunkt V zum Koordinatensystem I ist genau identische wie die Beziehung zwischen dem Punkt K zum Koordinatensystem J, d.h.

$$ \vec{B}(K) = \begin{bmatrix} 0 \\ B_2(V) \\ B_3(V) \end{bmatrix}_{\underline{\vec{\jmath}}} = \begin{bmatrix} 0 \\ B_2(V) \\ B_3(V) \end{bmatrix}^{\mathrm{T}} \cdot \underline{\vec{\jmath}} $$

Rechnen wir die MFS von J-Koordinaten ins Magnetkoordinaten I zurück, erhalten wir

$$ \begin{aligned} \vec{B}(K) &= \begin{bmatrix} 0 \\ B_2(V) \\ B_3(V) \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} \cos\theta_k & \sin\theta_k & 0 \\ -\sin\theta_k & \cos\theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \underline{\vec{\imath}} \\ &= \begin{bmatrix} -B_2(V)\sin\theta_k \\ B_2(V)\cos\theta_k \\ B_3(V) \end{bmatrix}_{\underline{\vec{\imath}}} \end{aligned} $$

Jetzt können wir die MFS an jedem Punkt berechnen.

Anmerkung: Um Rechenzeit zu sparen, rechnen wir die Magnetfeldstärke schon vorher. D.h. Der Vergleichspunkt fällt vielleicht nicht auf einem Knoten, an dem die Magnetfeldstärke berechnet wurde. Zur Erhöhung der Genauigkeit wird zudem eine 2D Interpolation (der numerisch berechneten MFDs) durchgeführt:

$$ B(y,z) = \frac{1}{(v_{y2} - v_{y1})(v_{z2} - v_{z1})} \begin{bmatrix} v_{y2} - y & y - v_{y1} \end{bmatrix} \begin{bmatrix} B_{y1z1} & B_{y1z2} \\ B_{y2z1} & B_{y2z2} \end{bmatrix} \begin{bmatrix} v_{z2} - z \\ z - v_{z1} \end{bmatrix} $$

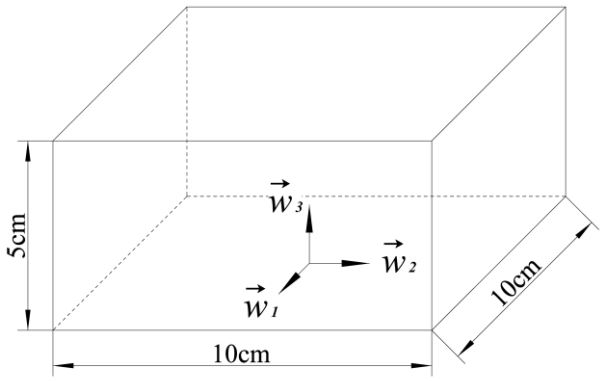
3 Sensoren und Sensorarray

3.1 Workspace Bilden

Annahme: Workspace $10 \times 10 \times 5$ cm^3

Anzahl der Sensoren jeder Seite: 5

Bildung der Weltkoordinaten $\underline{\vec{w}}$ im Workspace:



$\underline{\vec{w}}$ liegt an dem Mittelpunkt auf der Unterfläche im Arbeitsraum.

Die Sensoren werden gleichmäßig an der Ebene $\overrightarrow{w_1} - \overrightarrow{w_2}$ eingesetzt. Notation der Sensorstelle:

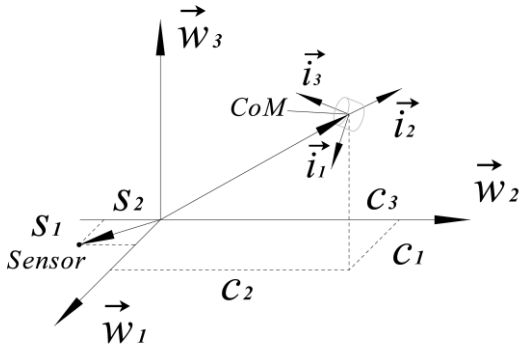
$$^{W}\vec{r}^{S_i} = [s_{i,1} \quad s_{i,2} \quad 0] \cdot \underline{\vec{w}}$$

3.2 Magnetorientierung

3.2.1 Beschreibung der Magnetorientierung

Magnetorientierung bedeutet die Positions- und Rotationszustand des Magnets. Das Projekt bezieht sich genau auf der Bestimmung der Magnetorientierung.

Translationsberechnen



Der Schwerpunkt (Mittelpunkt, oder Ursprung der Magnetkoordinate) vom Magnet C ist

$$^{W}\vec{r}^{C} = [c_1 \quad c_2 \quad c_3] \cdot \underline{\vec{w}}$$

Der Sensor Stelle ist

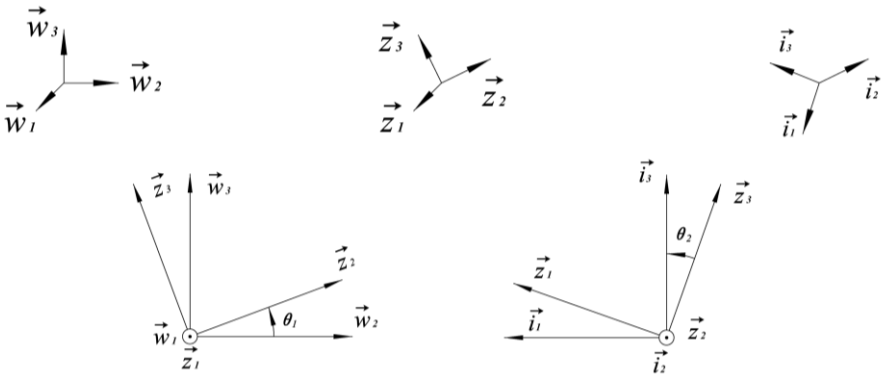
$$^{W}\vec{r}^{S_i} = [s_{i,1} \quad s_{i,2} \quad 0] \cdot \underline{\vec{w}}$$

Dann folgt

$$^{C}\vec{r}^{S_i} = -^{W}\vec{r}^{C} + {}^{W}\vec{r}^{S_i} = \begin{bmatrix} -c_1 + s_{i,1} \\ -c_2 + s_{i,2} \\ -c_3 \end{bmatrix}^{\mathrm{T}} \cdot \underline{\vec{w}}$$

Rotationsberechnen

Da der Magnet symmetrisch um $\vec{\imath_3}$ Achse ist, besitzt der Magnet nur 2 Freiheitsgrad bei der Rotation. Definiere ich jetzt die Rotationsreihfolge: erst um $\vec{\imath_1}$ (oder $\vec{w_1}$) (nach der Rotation ergibt sich das Koordinatensystem Z), danach um $\vec{\imath_2}$ (oder $\vec{z_2}$).



kann man direkt erhalten

$$\underline{\vec{w}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_1 & -\sin\theta_1 \\ 0 & \sin\theta_1 & \cos\theta_1 \end{bmatrix} \underline{\vec{z}}$$

$$\underline{\vec{z}} = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ 0 & 1 & 0 \\ -\sin\theta_2 & 0 & \cos\theta_2 \end{bmatrix} \underline{\vec{\imath}}$$

Ferner

$$^{W}\underline{m}^{I} = {}^{W}\underline{m}^{Z} \cdot {}^{Z}\underline{m}^{I} = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ \sin\theta_1 \sin\theta_2 & \cos\theta_1 & -\sin\theta_1 \cos\theta_2 \\ -\cos\theta_1 \sin\theta_2 & \sin\theta_1 & \cos\theta_1 \cos\theta_2 \end{bmatrix}$$

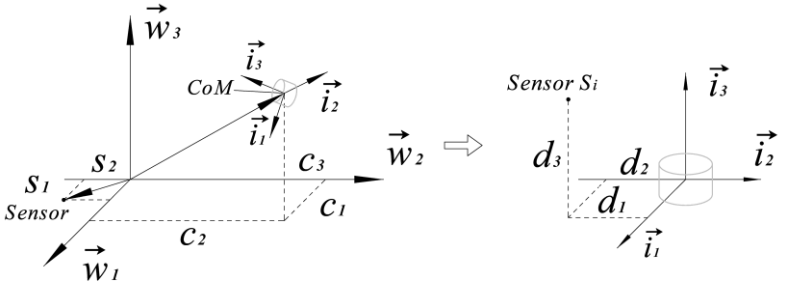
$$^{I}\underline{m}^{W} = \left({}^{W}\underline{m}^{I}\right)^{\mathrm{T}} = \begin{bmatrix} \cos\theta_2 & \sin\theta_1 \sin\theta_2 & -\cos\theta_1 \sin\theta_2 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ \sin\theta_2 & -\sin\theta_1 \cos\theta_2 & \cos\theta_1 \cos\theta_2 \end{bmatrix}$$

3.2.2 Koordinaten-Transformation zwischen Sensoren und Magnet

Fassen wir die Translation (relative Position zu einem Sensor) und Rotation zusammen

$$\begin{aligned} {}^{C}\vec{r}^{S_i} &= \begin{bmatrix} -c_1 + s_{i,1} \\ -c_2 + s_{i,2} \\ -c_3 \end{bmatrix}^{\mathrm{T}} \cdot \underline{\vec{w}} = \begin{bmatrix} -c_1 + s_{i,1} \\ -c_2 + s_{i,2} \\ -c_3 \end{bmatrix}^{\mathrm{T}} \cdot {}^{W}\underline{m}^{I} \cdot \underline{\vec{\imath}} \\ &= \begin{bmatrix} -c_1 + s_{i,1} \\ -c_2 + s_{i,2} \\ -c_3 \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 \\ \sin\theta_1 \sin\theta_2 & \cos\theta_1 & -\sin\theta_1 \cos\theta_2 \\ -\cos\theta_1 \sin\theta_2 & \sin\theta_1 & \cos\theta_1 \cos\theta_2 \end{bmatrix} \cdot \underline{\vec{\imath}} \\ &= \begin{bmatrix} \left(-c_1 + s_{i,1}\right)\cos\theta_2 + \left(-c_2 + s_{i,2}\right)\sin\theta_1 \sin\theta_2 + c_3 \cos\theta_1 \sin\theta_2 \\ \left(-c_2 + s_{i,2}\right)\cos\theta_1 - c_3 \sin\theta_1 \\ \left(-c_1 + s_{i,1}\right)\sin\theta_2 - \left(-c_2 + s_{i,2}\right)\sin\theta_1 \cos\theta_2 - c_3 \cos\theta_1 \cos\theta_2 \end{bmatrix}_{\vec{\imath}} =: \begin{bmatrix} d_{i,1} \\ d_{i,2} \\ d_{i,3} \end{bmatrix}_{\vec{\imath}} \end{aligned}$$

also



Wir wissen schon, dass das Magnetfeld am Punkt $S_i = [d_{i,1} \quad d_{i,2} \quad d_{i,3}]_{\vec{I}}$ ist (Sieh Kapitel 2.4):

$$ {}^{I}\vec{r}^{S_i'} = \begin{bmatrix} 0 \\ -d_{i,1}\sin\theta_k + d_{i,2}\cos\theta_k \\ d_{i,3} \end{bmatrix}_{\underline{\vec{I}}} =: \begin{bmatrix} 0 \\ d'_{i,2} \\ d'_{i,3} \end{bmatrix} $$

und

$$ \vec{B}({}^{C}\vec{r}^{S_i}) = \begin{bmatrix} -B_2(S_i')\sin\theta_k \\ B_2(S_i')\cos\theta_k \\ B_3(S_i') \end{bmatrix}_{\underline{\vec{I}}} $$

mit $\theta_k = -\operatorname{atan}\frac{d_{i,1}}{d_{i,2}} + \delta\cdot\pi$, und $\delta = \begin{cases} -1, \text{für } k_2 \le 0, k_1 \ge 0 \\ 0, \quad \text{für } k_2 > 0 \\ 1, \quad \text{für } k_2 \le 0, k_1 < 0 \end{cases}$

Rechnen wir die MFS an Sensor i in den Weltkoordinaten um:

$$ \begin{aligned} \vec{B}({}^{C}\vec{r}^{S_i}) &= \begin{bmatrix} -B_2(S_i')\sin\theta_k \\ B_2(S_i')\cos\theta_k \\ B_3(S_i') \end{bmatrix}_{\underline{\vec{I}}} = \begin{bmatrix} -B_2(S_i')\sin\theta_k \\ B_2(S_i')\cos\theta_k \\ B_3(S_i') \end{bmatrix}^{\mathrm{T}} \cdot \underline{\vec{I}} \\ &= \begin{bmatrix} -B_2(S_i')\sin\theta_k \\ B_2(S_i')\cos\theta_k \\ B_3(S_i') \end{bmatrix}^{\mathrm{T}} \cdot {}^{I}\underline{m}^{W} \cdot \underline{\vec{w}} \\ &= \begin{bmatrix} -B_2(S_i')\sin\theta_k \\ B_2(S_i')\cos\theta_k \\ B_3(S_i') \end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix} \cos\theta_2 & \sin\theta_1\sin\theta_2 & -\cos\theta_1\sin\theta_2 \\ 0 & \cos\theta_1 & \sin\theta_1 \\ \sin\theta_2 & -\sin\theta_1\cos\theta_2 & \cos\theta_1\cos\theta_2 \end{bmatrix} \cdot \underline{\vec{w}} \\ &= \begin{bmatrix} -B_2(S_i')\sin\theta_k\cos\theta_2 + B_3(S_i')\sin\theta_2 \\ -B_2(S_i')\sin\theta_k\sin\theta_1\sin\theta_2 + B_2(S_i')\cos\theta_k\cos\theta_1 - B_3(S_i')\sin\theta_1\cos\theta_2 \\ B_2(S_i')\sin\theta_k\cos\theta_1\sin\theta_2 + B_2(S_i')\cos\theta_k\sin\theta_1 + B_3(S_i')\cos\theta_1\cos\theta_2 \end{bmatrix}_{\vec{w}} \end{aligned} $$

mit $S_i' = \begin{bmatrix} 0 \\ -d_{i,1}\sin\theta_k + d_{i,2}\cos\theta_k \\ d_{i,3} \end{bmatrix}_{\vec{I}}$ der Vergleichspunkt vom Sensorposition in Magnetkoordinaten,

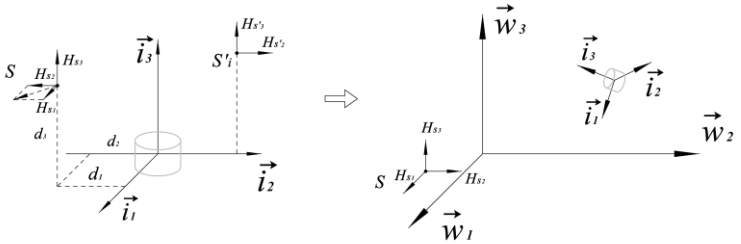
$\theta_k = -\operatorname{atan}\frac{d_{i,1}}{d_{i,2}} + \delta\cdot\pi$, und $\delta = \begin{cases} -1, \text{für } k_2 \le 0, k_1 \ge 0 \\ 0, \quad \text{für } k_2 > 0 \\ 1, \quad \text{für } k_2 \le 0, k_1 < 0 \end{cases}$,

$\begin{bmatrix} d_{i,1} \\ d_{i,2} \\ d_{i,3} \end{bmatrix}_{\vec{I}} = \begin{bmatrix} \left(-c_1 + s_{i,1}\right)\mathrm{c}\,\theta_2 + \left(-c_2 + s_{i,2}\right)\mathrm{s}\,\theta_1\,\mathrm{s}\,\theta_2 + c_3\,\mathrm{c}\,\theta_1\,\mathrm{s}\,\theta_2 \\ \left(-c_2 + s_{i,2}\right)\mathrm{c}\,\theta_1 - c_3\,\mathrm{s}\,\theta_1 \\ \left(-c_1 + s_{i,1}\right)\mathrm{s}\,\theta_2 - \left(-c_2 + s_{i,2}\right)\mathrm{s}\,\theta_1\,\mathrm{c}\,\theta_2 - c_3\,\mathrm{c}\,\theta_1\,\mathrm{c}\,\theta_2 \end{bmatrix}_{\vec{I}}$ die Position vom S_i,

$[c_1 \quad c_2 \quad c_3 \quad \theta_1 \quad \theta_2]_{\vec{w}}^{\mathrm{T}}$ die Orientation vom Magnet in Weltkoordinaten

$[0 \quad s_{i,1} \quad s_{i,2}]_{\vec{w}}^{\mathrm{T}}$ die Position vom Sensor S_i in Weltkoordinaten

Das folgende Bild zeigt das Verfahren:



Jetzt können wir Magnetfeldstärke an jeder Stelle bei jeder Position und Orientation des Magnets in Weltkoordinaten ausdrücken.

3.3 Sensoren

3.3.1 Sensoren und ADC

Wir nutzen hier Hall-Sensoren, um dir Magnet Fluss Dichte zu messen. (anstatt Magnet Field Stärke).

Sensoren werden normalerweise mit AD-Wandler zusammengenutzt. Es gibt einige wichtige Daten von ihr:

Range

Als gezeigt, die MFD verändert sich sehr stark. In der Nähe vom Magnet ist die MFD größer als 10000 Gauß. Aber sie sinkt sehr schnell auf ca. 10 Gauß mit der Entfernung von 5cm.

Somit entsteht ein Problem: Wenn wir die MFD an allen Punkten detektieren möchten, müssen wir die Sensoren mit sehr großem Rang wählen, aber es wird mit sich bringen, dass die Auflösung im Bereich außerhalb 5 cm sehr schlimm ist.

Abhilfe: Beschränkung der Position des Magnets, um zu vermeiden, dass die Sensoren zu groß MFD (außerhalb der Range) detektieren. (also wähle einen gültigen Bereich, sieh unten)

Drift

Drift ist hier der Fehler von der Linearität der Kennlinien des Sensors. Das ist eigentlich die wichtigste Kenngröße eines Sensors.

Sensitivität

Änderung der Ausgangsspannung bei der Änderung von MFD.

ADC Auflösung

Bestimmt den Quantisierungsfehler. ADC-Auflösung und Sensitivität haben starke Beziehung.

3.3.2 Gültigerer Bereich

Es gibt 2 Arten nicht-gültiges Bereichs

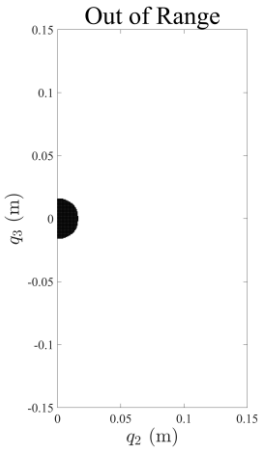
1. Außerhalb der Range

Wir müssen die Range von einem Sensor betrachten. Normalerweise ist die Range von Hallsensor $[B_{max}, B_{min}]$

Wählen wir den Bereich, in dem die MFD (Magnetflussdichte) höher als 500 Gauß ist,

$$\begin{pmatrix} q_2 \\ q_3 \end{pmatrix} \in \left\{ \underline{q} : B_m\left(\underline{q}\right) \geq 500 \right\}$$

erhalten wir:



D.h. Der Magnet soll nicht innerhalb 15 mm von dem Sensor sein. (Aber das muss nicht immer erfüllt werden, in der Lokalisierungsalgorithmus werden die Signale, die außerhalb der Range, übersehen.)

2. Kleiner als die Auflösung

Wenn der A/D Wandler k-bit ist und Eingangsspannung von 0 bis 5 V ist. Ist die Spannungsauflösung (Quantisierungsstufe):

$$\Delta V = \frac{5}{2^k}$$

D.h. Nur die Spannungsänderung, die größer als ΔV ist, kann detektiert werden. Dementsprechend soll die Änderung von MFD größer als

$$\Delta B_m = \frac{\Delta V}{S} = \frac{5}{2^k S}$$

wobei S Sensitivität

Es ist offenbar, dass ADC Auflösung und Sensitivität starte Beziehung haben. Also wenn wir die Sensoren mit schlechter Sensitivität haben, können wir ADC mit höherer Auflösung zu wählen, damit der Nachteil ausgeglichen wird.

Wenn wir mindestens a Meter Positionsänderung detektieren möchten, brauchen wir mindestens

$$\frac{\partial B_m}{\partial q} \cdot a > \Delta B_m = \frac{5}{2^k S}$$

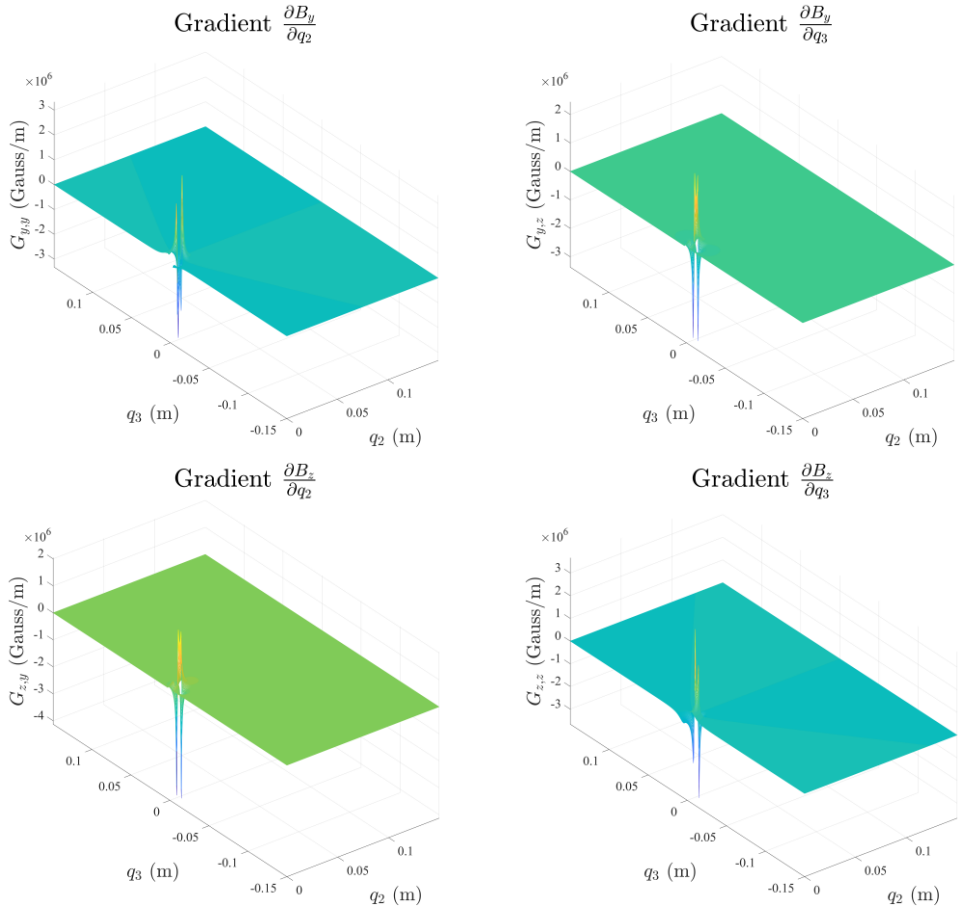
$$\frac{\partial B_m}{\partial q} > \frac{5}{2^k \cdot S \cdot a}$$

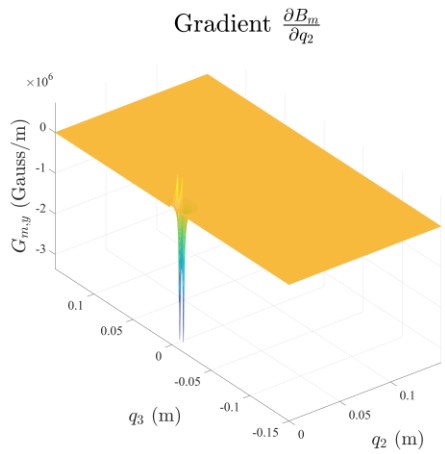
wobei $\frac{\partial B_m}{\partial q}$ die Gradienten der MFD

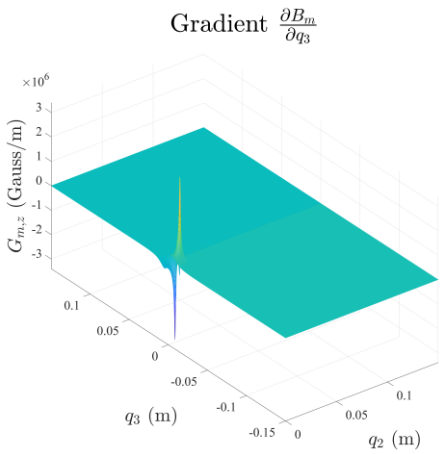
Die Ableitung des MFD kann man mit Finite Differenz Methode berechnen, also

$$\nabla B_M(q_2, q_3) = \begin{bmatrix} \frac{\partial B_M}{\partial q_2} \\ \frac{\partial B_M}{\partial q_3} \end{bmatrix} \approx \begin{bmatrix} \frac{B_M(q_2 + \Delta q_2, q_3) - B_M(q_2, q_3)}{\Delta q_2} \\ \frac{B_M(q_2, q_3 + \Delta q_3) - B_M(q_2, q_3)}{\Delta q_3} \end{bmatrix}$$

erhalten wir



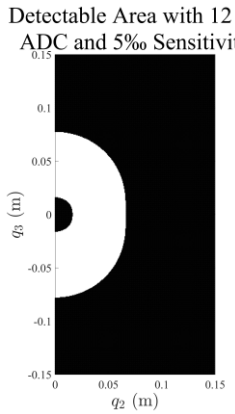


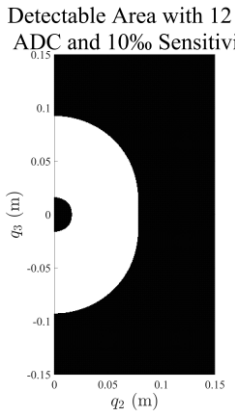


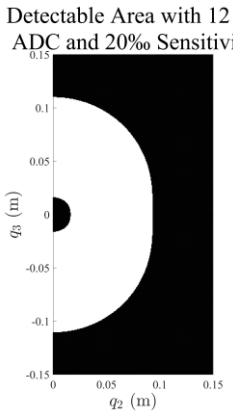
Wir möchten nun 1 mm Bewegung detektieren, also

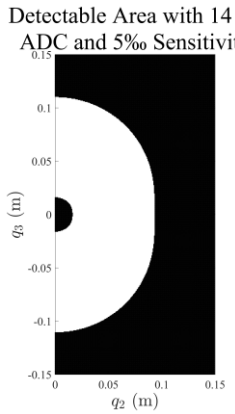
$$a = 1\ \text{mm} = 0.001\ \text{m}$$

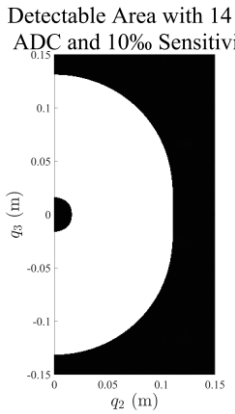
erhalten wir

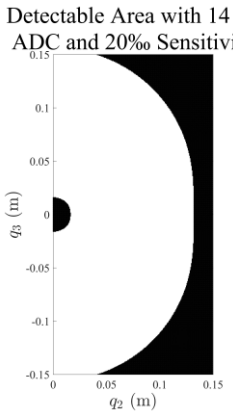


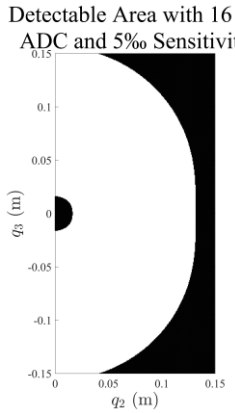


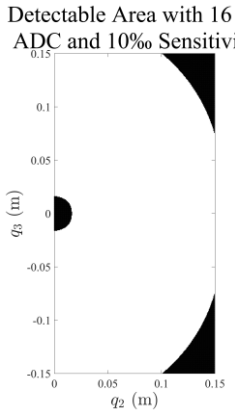


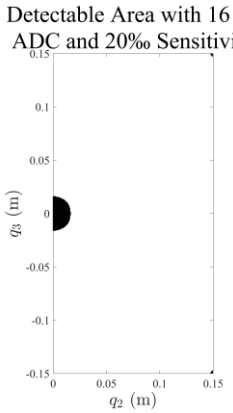


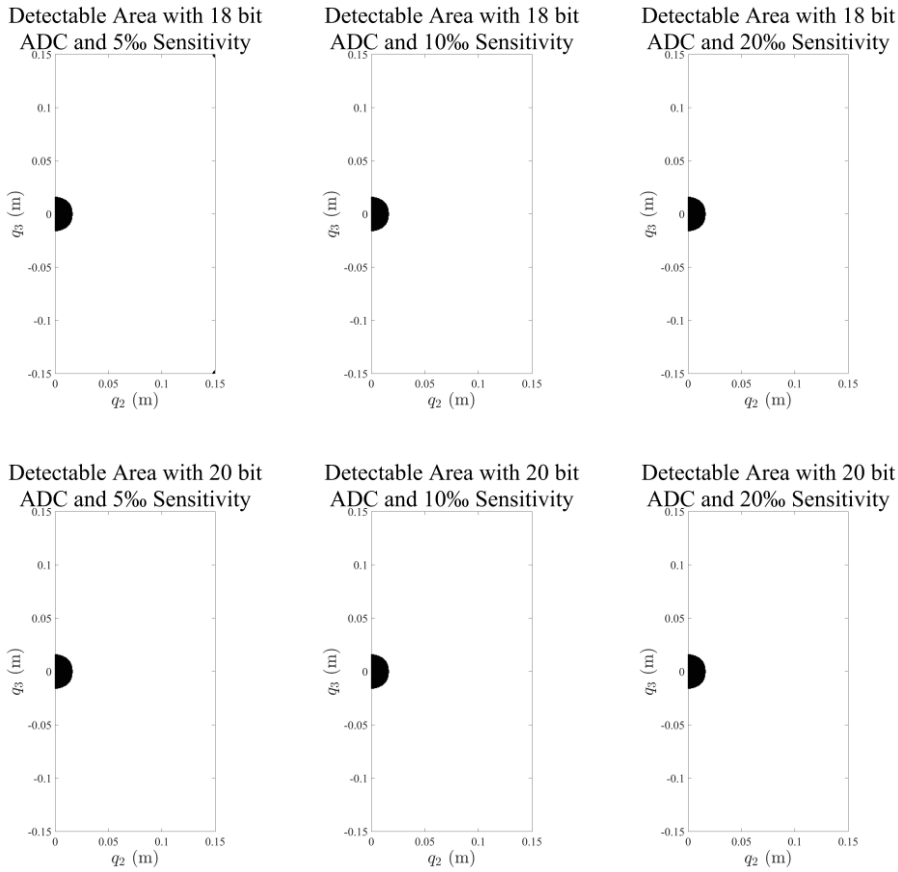












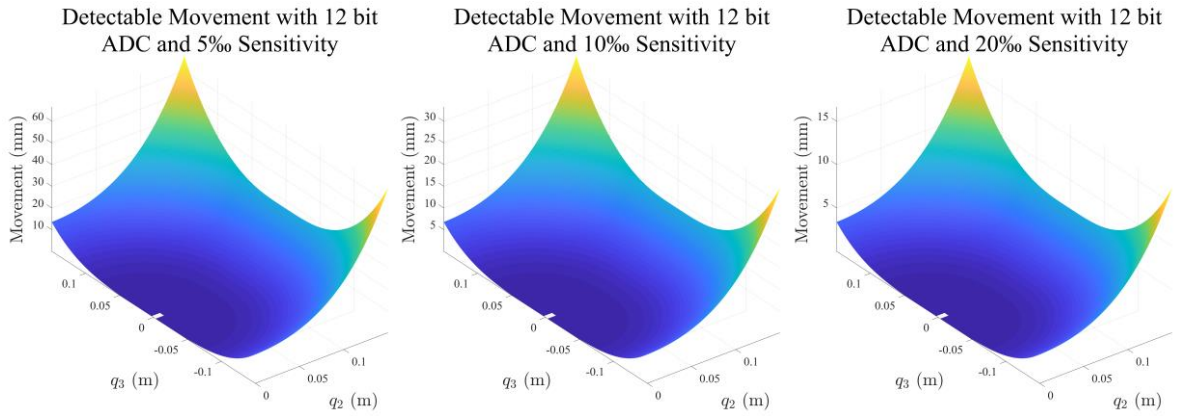
16 bit ADC und eine Sensitivität von 10 %0 oder höher. Unter diesem Umständen ist die Bewegung innerhalb des hier weiß dargestellten Bereiches detektierbar

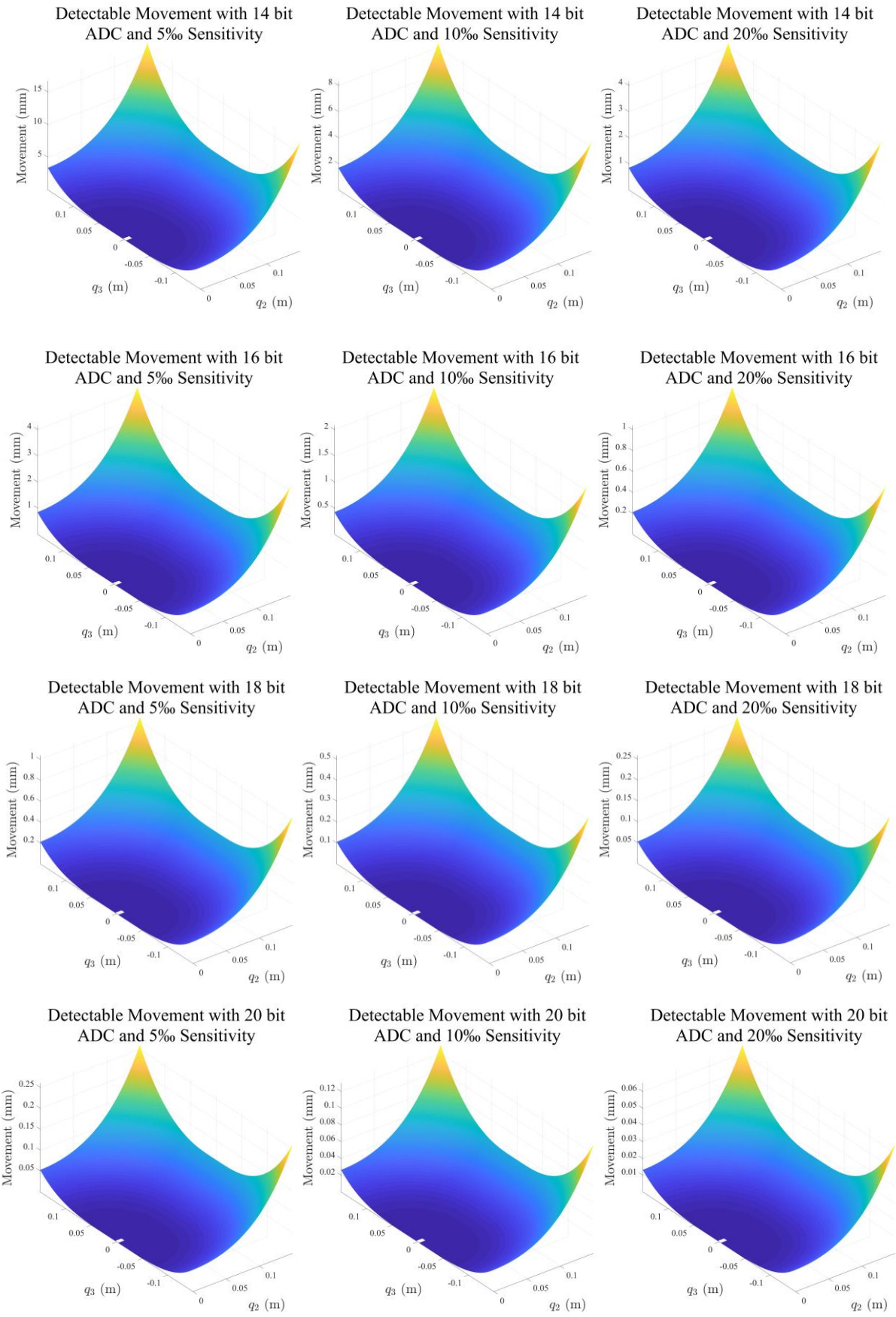
Da die Sensoren mit höherer Sensitivität teuer sind, wählen wir am besten 18-bit ADC und 10 mV/G Sensitivität. Oder 20-bit ADC und 5 mV/G Sensitivität.

Aus der obergenannten Formel wird es auch hergeleitet, dass

$$a > \frac{5}{2^k \cdot S \cdot \frac{\partial B_m}{\partial q_2}}$$

dann erhalten wir die detektierbare Bewegungen a unter verschiedene Sensordaten:




Detectable Movement with 14 bit ADC and 5‰ Sensitivity
Detectable Movement with 14 bit ADC and 10‰ Sensitivity
Detectable Movement with 14 bit ADC and 20‰ Sensitivity
Detectable Movement with 16 bit ADC and 5‰ Sensitivity
Detectable Movement with 16 bit ADC and 10‰ Sensitivity
Detectable Movement with 16 bit ADC and 20‰ Sensitivity
Detectable Movement with 18 bit ADC and 5‰ Sensitivity
Detectable Movement with 18 bit ADC and 10‰ Sensitivity
Detectable Movement with 18 bit ADC and 20‰ Sensitivity
Detectable Movement with 20 bit ADC and 5‰ Sensitivity
Detectable Movement with 20 bit ADC and 10‰ Sensitivity
Detectable Movement with 20 bit ADC and 20‰ Sensitivity
Movement (mm)
q_3 (m)
q_2 (m)

3.3.3 Sensor Modellierung

Der Ausgang eines Sensors ist

$$V_n = B_s \cdot S \cdot (1 + e_s) + V_z$$

wobei V Ausgangsspannung

B_s MFD an dem Sensor

e_s Sensitivitätsfehler

V_z Nullausgang

Ausgang eines Quantizers (AD Wandlers) ist

$$\frac{5}{2^k} n < V_n < \frac{5}{2^k}(n+1)$$

$$n = \left\lfloor V_n * \frac{2^k}{5} \right\rfloor$$

$$V_q = \frac{5}{2^k} n = \frac{5}{2^k} \left\lfloor V_n * \frac{2^k}{5} \right\rfloor$$

Beobachteter MFD

$$B_b = \frac{(V_q - V_z)}{S}$$

Mathematisches Modell:

Bei der Simulation, generieren wir beliebig (oder mit anderen Strategien) eine Orientierung des Magnets. Und dann berechnete der MFD an jedem Sensor. Danach um das Sensorsystem zu simulieren, sollen wir das MFD ins Spannung umrechnen, verrauschen, quantisieren, und schließlich ins MFD zurückrechnen.

$$V_n = B_s \cdot S \cdot (1 + rand(-1{,}1) \times e_s) + V_z$$

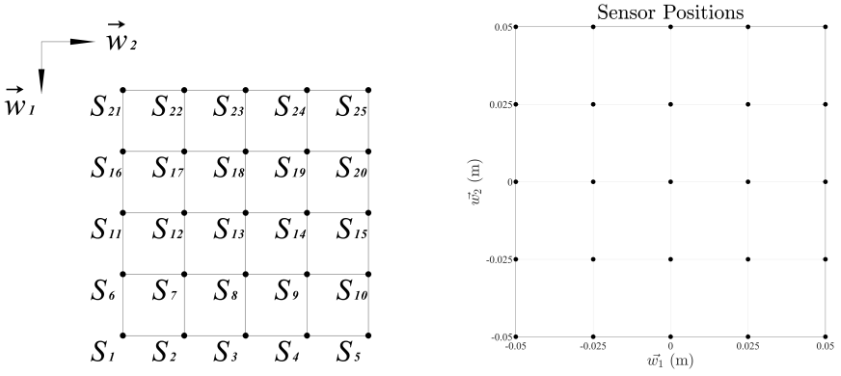
$$V_q = \left\lfloor \left(V_n + \frac{5}{2^{k+1}} \right) * \frac{2^k}{5} \right\rfloor \times \frac{5}{2^k}$$

$$B_b = \frac{(V_q - V_z)}{S}$$

3.4 Sensorarray

3.4.1 Aufbau

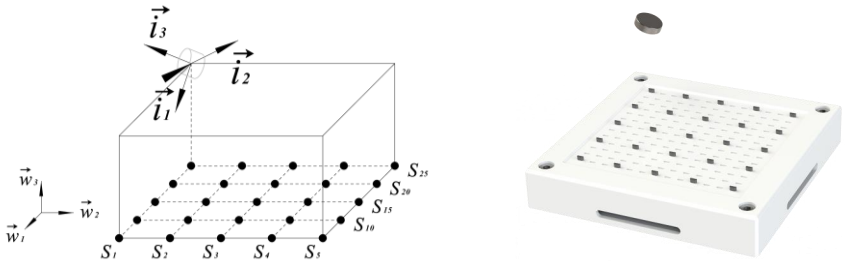
Der Sensorarray wird so entworfen, dass es aus 5×5 gleichverteilte Sensoren besteht.



3.4.2 Sensorarray Analyse

Um quantifizieren bzw. abschätzen zu können auf welche Distanz wir einen Magneten detektieren, wird der Magnet zunächst an einer extremen Stelle gesetzt, also

$$C = [-0.05 \quad -0.05 \quad 0.05 \quad \theta_1 \quad \theta_2]^{\mathrm{T}}_{\vec{w}}$$



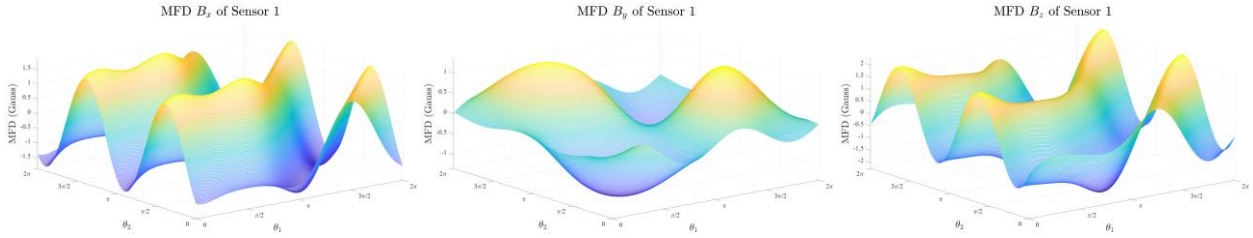
Betrachtet werden die Änderungen der Magnetfeldstärke bei den Änderungen von θ_1 und θ_2 an jeder Sensorstelle.

Hier werden nur die Magnetfeld Änderungen an Sensoren, die an wichtigen Stellen legen, also Sensor 5, 11, 21, gezeigt.

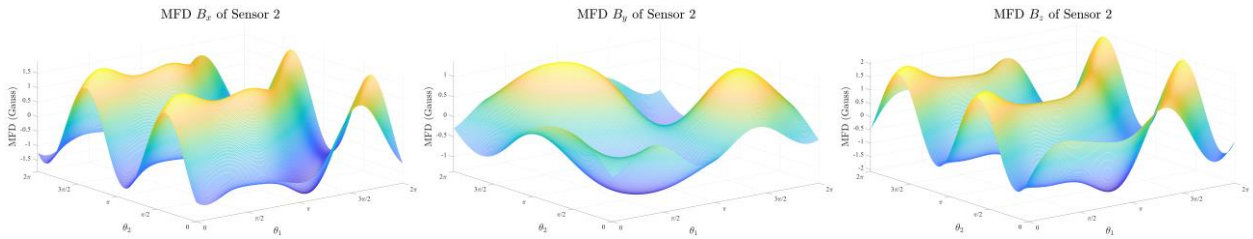
Grund für Sensor 5 und 21 ist, dass sie das schwächste bzw. stärkste Signale aufnehmen. Um 5 Parameter vom Magnet zu bestimmen, sind mindestens 5 Sensoren benötigt, die das Signal detektieren können. Natürlich ist die beste Situation, dass alle Sensoren das Magnetfeld vom Magnet detektieren können, damit können wir möglichst viele Informationen gewinnen und benutzen.

Ergebnisse sind folgend:

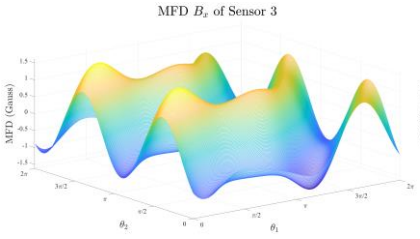
Sensor 1

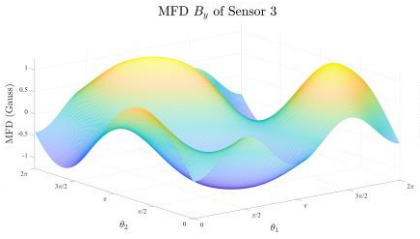


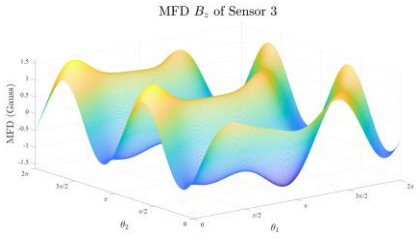
Sensor 2



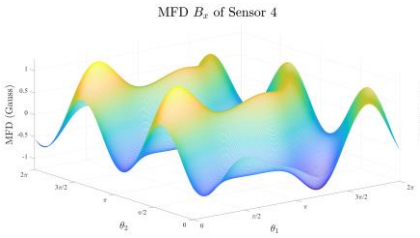
Sensor 3

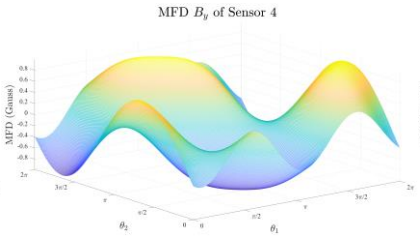


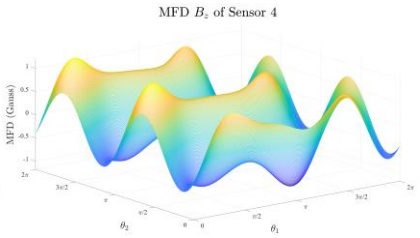




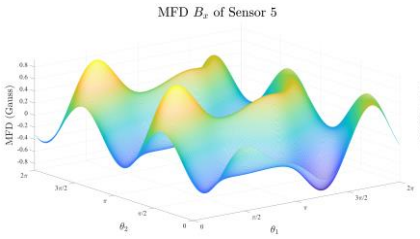
Sensor 4

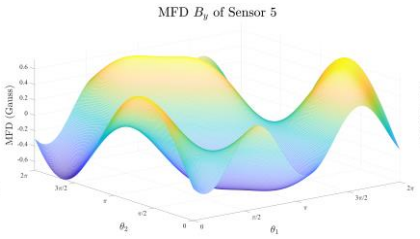


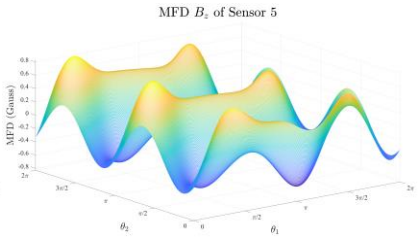




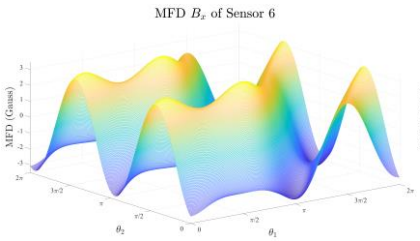
Sensor 5

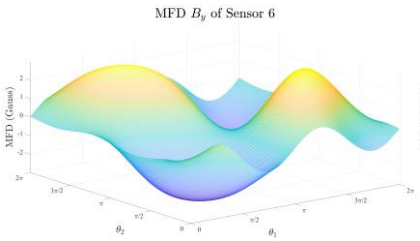


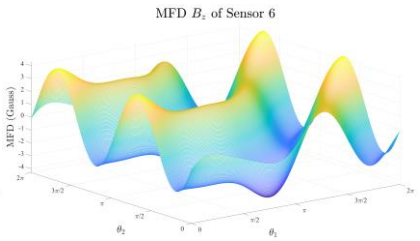




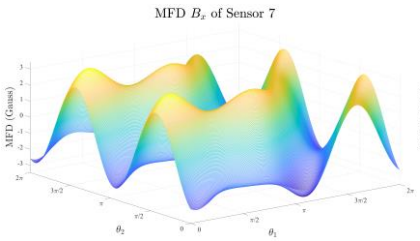
Sensor 6

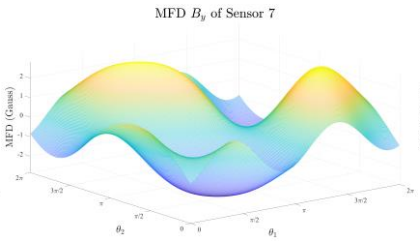


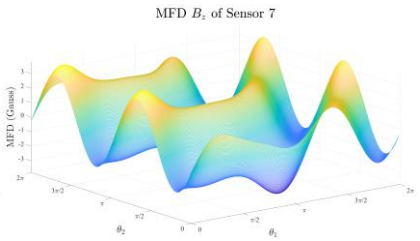




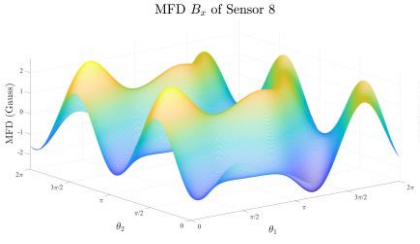
Sensor 7

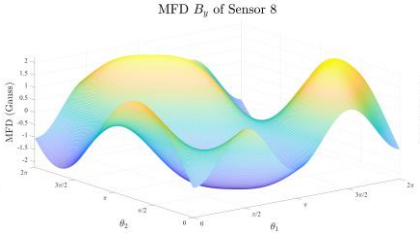


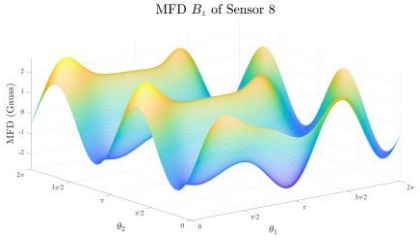




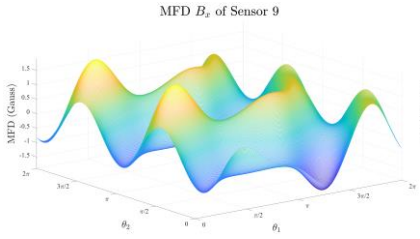
Sensor 8

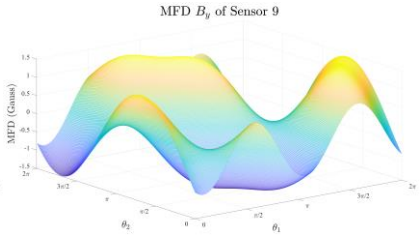


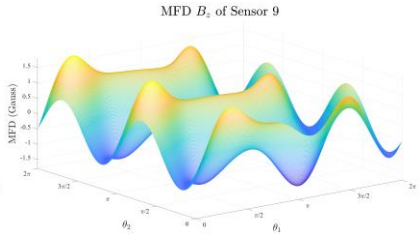




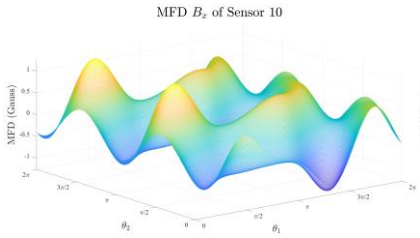
Sensor 9

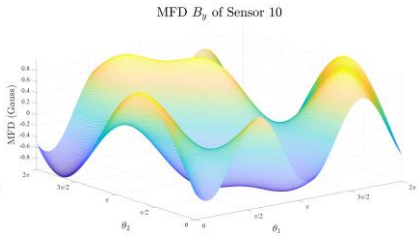


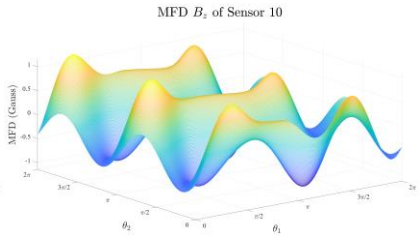




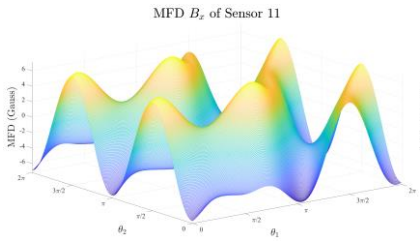
Sensor 10

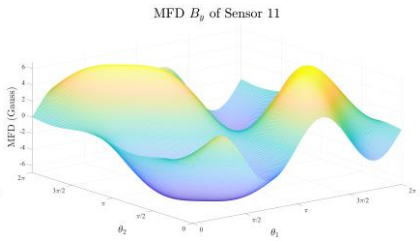


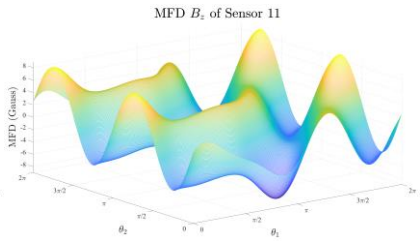




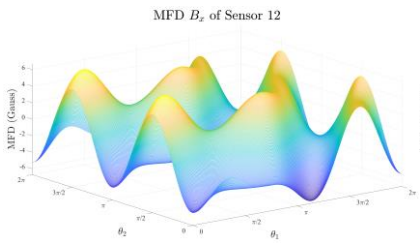
Sensor 11

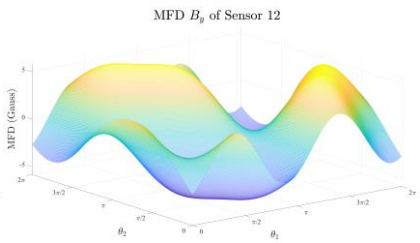


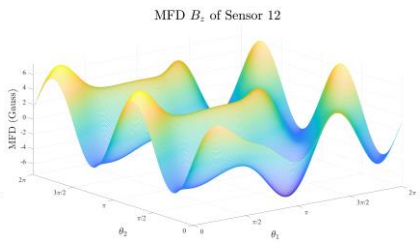




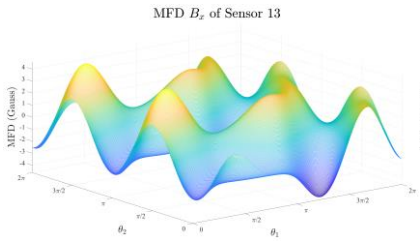
Sensor 12

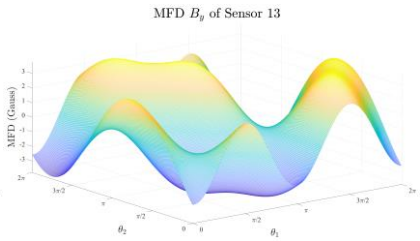


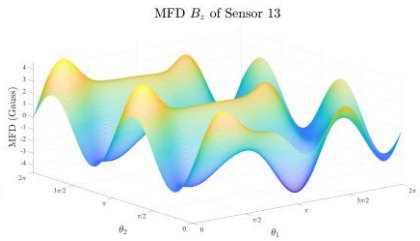




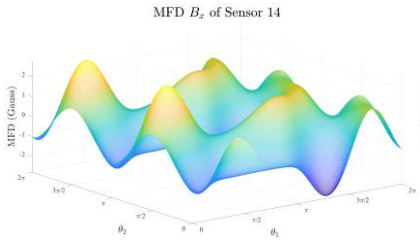
Sensor 13

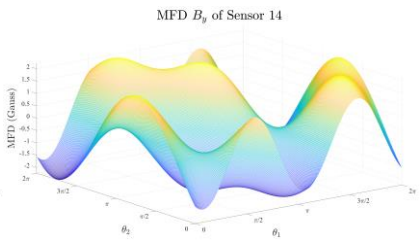


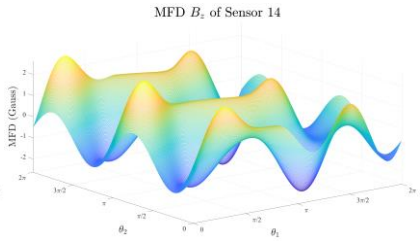




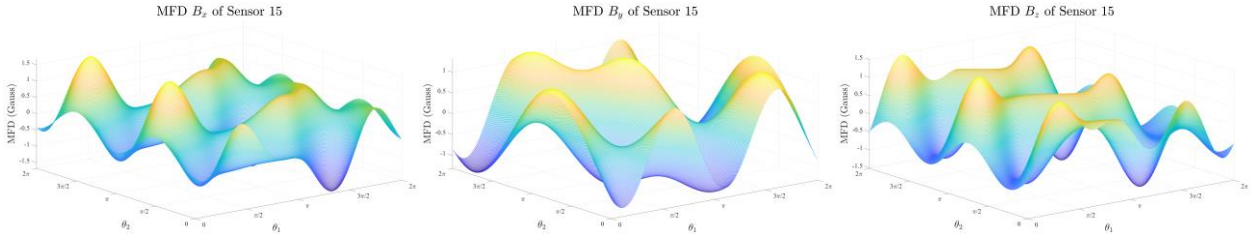
Sensor 14



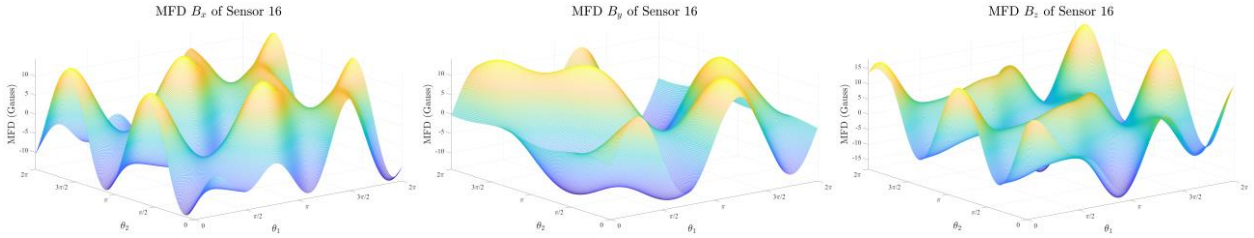




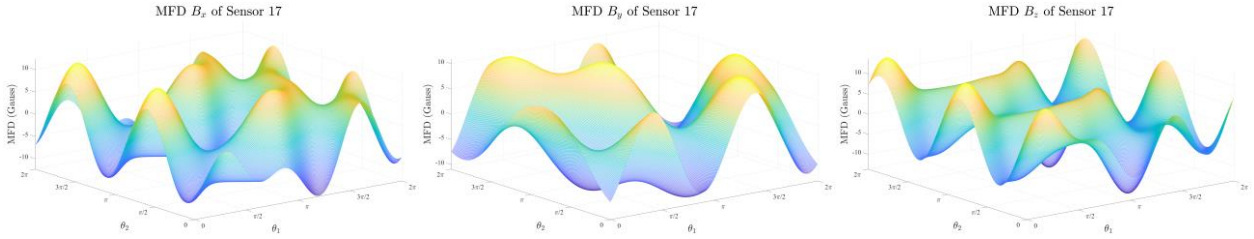
Sensor 15



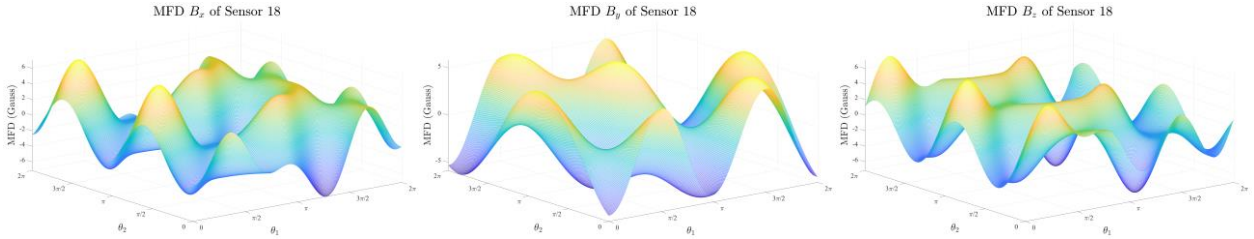
Sensor 16



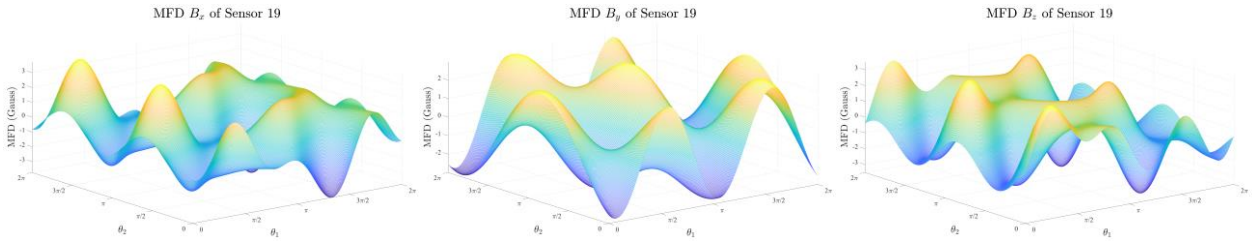
Sensor 17



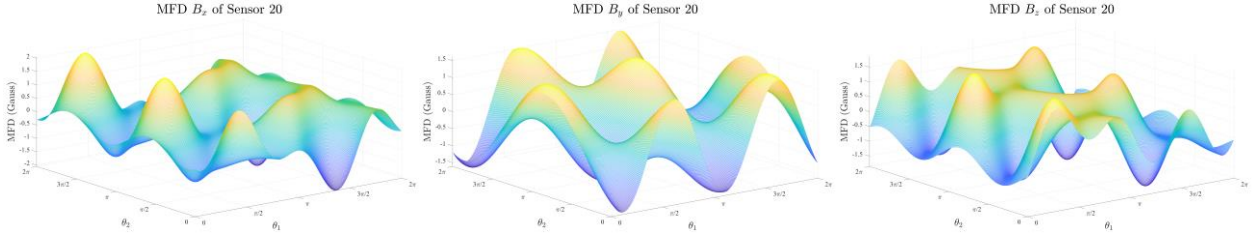
Sensor 18



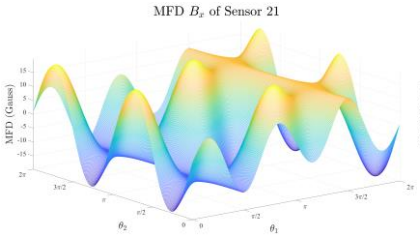
Sensor 19

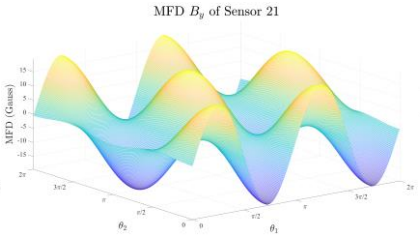


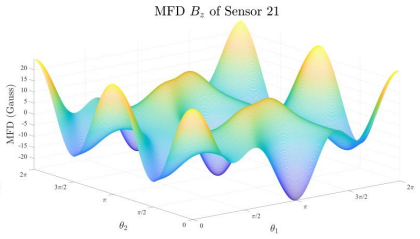
Sensor 20



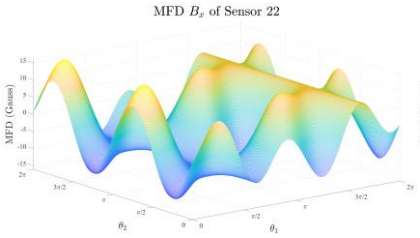
Sensor 21

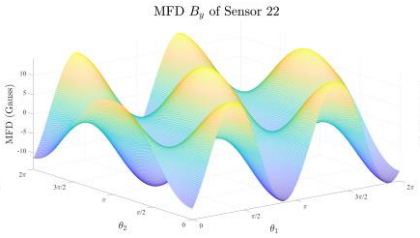


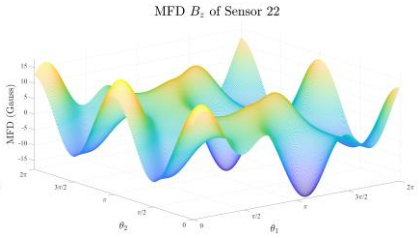




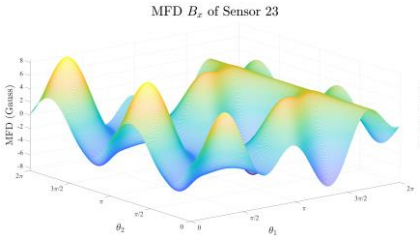
Sensor 22

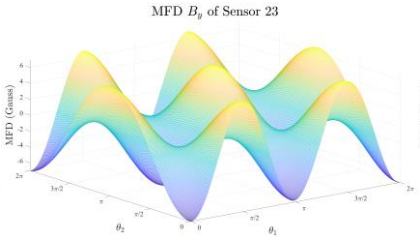


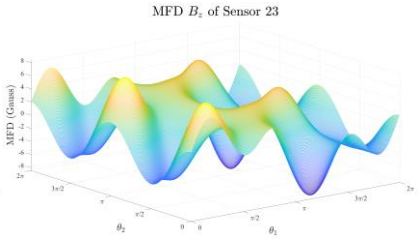




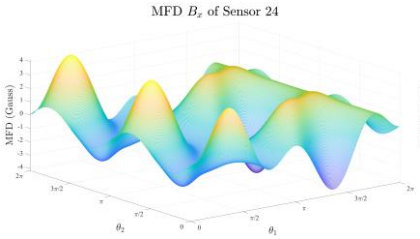
Sensor 23

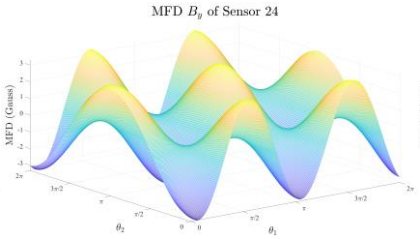


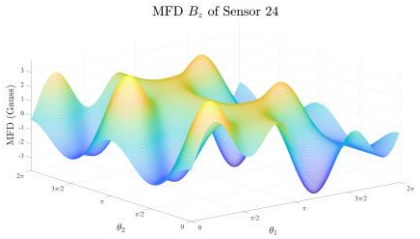




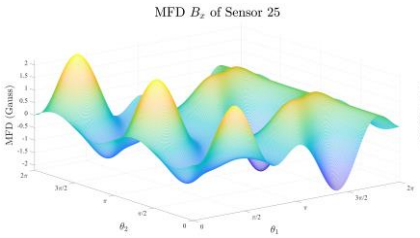
Sensor 24

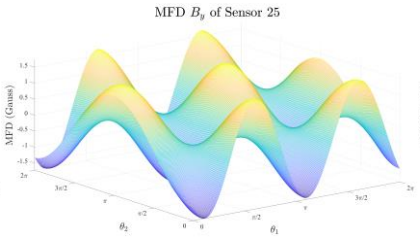


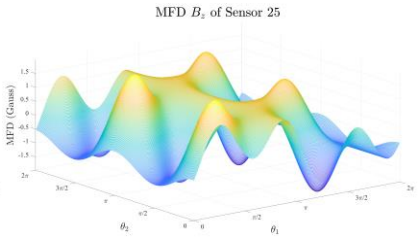




Sensor 25







4 Lokalisierungsalgorithmen

Aufforderung

lokalisiere jede 10 ms.

Annahme

max. Geschwindigkeit 1m/s

max. Rotationsgeschwindigkeit π/s

4.1 Mathematisches Modell

Das kinematische Modell des Softroboters ist in Momentan nicht verfügbar, deswegen können wir nur die Informationen aus den Sensoren benutzen. Ich nutze Maximum-Likelihood-Schätzer:

$$\hat{O}_{ML}(B_b) = \underset{O}{\operatorname{argmax}}\{f(B_b|O)\}$$

wobei $\hat{O}_{ML}$ die geschätzte Orientierung des Magnets

B_b die beobachtete MFD

O die Orientierung des Magnets

Um das Problem zu lösen, konvertierte ich es in einem Optimierungsproblem

$$\begin{aligned} \min G(O) &= \sum_{i=1}^{25} \frac{1}{2} \left\| B_{b,i,x} - B_{i,x}(O) \right\|^2 \\ \text{s.t. } & O_1 \in [-0.05, 0.05] \\ & O_2 \in [-0.05, 0.05] \\ & O_3 \in [0, 0.05] \end{aligned}$$

O_4 und O_5 sind in Prinzip in dem Intervall $[0, 2\pi]$. Aber wir können sie als unbegrenzte Variable behandeln.

4.2 Gradienten Methode

4.2.1 Suchrichtung

Die Suchrichtung $\vec{p}(O^{(n)})$ entspricht der negativen Gradienten-Richtung. Mit Hilfe von Jakobi-Matrix kann der Richtung einfach berechnet werden.

$$\vec{p}(O^{(n)}) = -\nabla\left(\sum_{i=1}^{25} \frac{1}{2} \left\| B_{b,i,x} - B_{i,x}(O^{(n)}) \right\|^2\right) = -\begin{bmatrix} \frac{\partial \sum_{i=1}^{25} \frac{1}{2} \left\| B_{b,i,x} - B_{i,x}(O^{(n)}) \right\|^2}{\partial O_1} \\ \frac{\partial \sum_{i=1}^{25} \frac{1}{2} \left\| B_{b,i,x} - B_{i,x}(O^{(n)}) \right\|^2}{\partial O_2} \\ \frac{\partial \sum_{i=1}^{25} \frac{1}{2} \left\| B_{b,i,x} - B_{i,x}(O^{(n)}) \right\|^2}{\partial O_3} \\ \frac{\partial \sum_{i=1}^{25} \frac{1}{2} \left\| B_{b,i,x} - B_{i,x}(O^{(n)}) \right\|^2}{\partial O_4} \\ \frac{\partial \sum_{i=1}^{25} \frac{1}{2} \left\| B_{b,i,x} - B_{i,x}(O^{(n)}) \right\|^2}{\partial O_5} \end{bmatrix}$$

$$= \sum_{i=1}^{25} \left[B_{b,i,x} - B_{i,x}(O^{(n)})\right] \nabla B_{i,x}(O^{(n)})$$

$$= \sum_{i=1}^{25} \left[B_{b,i,x} - B_{i,x}(O^{(n)})\right] \cdot \nabla d_i(O^{(n)}) \cdot \frac{\partial S_i'}{\partial d_i} \cdot \frac{\partial B}{\partial S_i'} \cdot \frac{\partial B_{i,x}}{\partial B}$$

$$= \sum_{i=1}^{25} \left[B_{b,i,x} - B_{i,x}\left(O^{(n)}\right)\right] \cdot \begin{bmatrix} \frac{\partial d_{i,1}}{\partial O_1^{(n)}} & \frac{\partial d_{i,2}}{\partial O_1^{(n)}} & \frac{\partial d_{i,3}}{\partial O_1^{(n)}} \\ \frac{\partial d_{i,1}}{\partial O_2^{(n)}} & \frac{\partial d_{i,2}}{\partial O_2^{(n)}} & \frac{\partial d_{i,3}}{\partial O_2^{(n)}} \\ \frac{\partial d_{i,1}}{\partial O_3^{(n)}} & \frac{\partial d_{i,2}}{\partial O_3^{(n)}} & \frac{\partial d_{i,3}}{\partial O_3^{(n)}} \\ \frac{\partial d_{i,1}}{\partial O_4^{(n)}} & \frac{\partial d_{i,2}}{\partial O_4^{(n)}} & \frac{\partial d_{i,3}}{\partial O_4^{(n)}} \\ \frac{\partial d_{i,1}}{\partial O_5^{(n)}} & \frac{\partial d_{i,2}}{\partial O_5^{(n)}} & \frac{\partial d_{i,3}}{\partial O_5^{(n)}} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial S'_{i,1}}{\partial d_{i,1}} & \frac{\partial S'_{i,2}}{\partial d_{i,1}} & \frac{\partial S'_{i,3}}{\partial d_{i,1}} \\ \frac{\partial S'_{i}}{\partial d_{i,2}} & \frac{\partial S'_{i,2}}{\partial d_{i,2}} & \frac{\partial S'_{i,3}}{\partial d_{i,2}} \\ \frac{\partial S'_{i}}{\partial d_{i,3}} & \frac{\partial S'_{i,2}}{\partial d_{i,3}} & \frac{\partial S'_{i,3}}{\partial d_{i,3}} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial B_1}{\partial S'_{i,1}} & \frac{\partial B_2}{\partial S'_{i,1}} & \frac{\partial B_3}{\partial S'_{i,1}} \\ \frac{\partial B_1}{\partial S'_{i,2}} & \frac{\partial B_2}{\partial S'_{i,2}} & \frac{\partial B_3}{\partial S'_{i,2}} \\ \frac{\partial B_1}{\partial S'_{i,3}} & \frac{\partial B_2}{\partial S'_{i,3}} & \frac{\partial B_3}{\partial S'_{i,3}} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial B_{i,x}}{\partial B_1} \\ \frac{\partial B_{i,x}}{\partial B_2} \\ \frac{\partial B_{i,x}}{\partial B_3} \end{bmatrix}$$

$$= \sum_{i=1}^{25} \left[B_{b,i,x} - B_{i,x}\left(O^{(n)}\right)\right] \cdot A \cdot \begin{bmatrix} 0 & -\sin\theta_k^{(n)} & 0 \\ 0 & \cos\theta_k^{(n)} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{\partial B_2}{\partial q_2} & \frac{\partial B_3}{\partial q_2} \\ 0 & \frac{\partial B_2}{\partial q_3} & \frac{\partial B_3}{\partial q_3} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ -\sin\theta_k^{(n)} \cos O_5^{(n)} \\ \sin O_5^{(n)} \end{bmatrix}$$

Wobei

$$A = \begin{bmatrix} -\cos O_5^{(n)} & 0 & -\sin O_4^{(n)} \\ -\sin O_4^{(n)} \sin O_5^{(n)} & -\cos O_4^{(n)} & \sin O_4^{(n)} \cos O_5^{(n)} \\ \cos O_4^{(n)} \sin O_5^{(n)} & -\sin O_4^{(n)} & -\cos O_4^{(n)} \cos O_5^{(n)} \\ \begin{pmatrix} \left(-O_2^{(n)} + s_{i,2}\right) \cos O_4^{(n)} \sin O_5^{(n)} - \\ O_3^{(n)} \sin O_4^{(n)} \sin O_5^{(n)} \end{pmatrix} & \begin{pmatrix} \left(O_2 - s_{i,2}\right) \sin O_4^{(n)} - \\ O_3^{(n)} \cos O_4^{(n)} \end{pmatrix} & \begin{pmatrix} \left(O_2^{(n)} - s_{i,2}\right) \cos O_4^{(n)} \cos O_5^{(n)} + \\ O_3^{(n)} \sin O_4^{(n)} \cos O_5^{(n)} \end{pmatrix} \\ \begin{pmatrix} \left(O_1^{(n)} - s_{i,1}\right) \sin O_5^{(n)} + \\ \left(-O_2^{(n)} + s_{i,2}\right) \sin O_4^{(n)} \cos O_5^{(n)} + \\ O_3^{(n)} \cos O_4^{(n)} \cos O_5^{(n)} \end{pmatrix} & 0 & \begin{pmatrix} \left(-O_1^{(n)} + s_{i,1}\right) \cos O_5^{(n)} + \\ \left(-O_2^{(n)} + s_{i,2}\right) \sin O_4^{(n)} \sin O_5^{(n)} + \\ O_3^{(n)} \cos O_4^{(n)} \sin O_5^{(n)} \end{pmatrix} \end{bmatrix}.$$

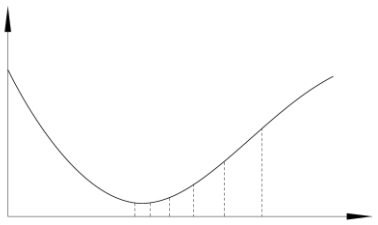
4.2.2 Schrittlänge

Iterative Suche nach dem Minimum Punkt mithilfe Backtracking-Algorithmus, also

$$\alpha_0 = \left\|\vec{p}\left(O^{(n)}\right)\right\|$$
$$\alpha_{k+1} = 0.8 \cdot \alpha_k$$

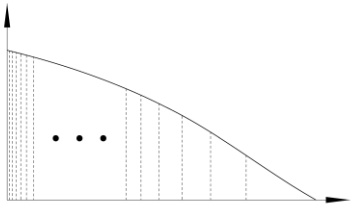
Endbedingung für Backtracking

1. Fall



$$G(O + \alpha_{k-1}) \geq G(O + \alpha_k) \leq G(O + \alpha_{k+1})$$
$$\alpha = \alpha_k$$

2. Fall



$$|\alpha_{k+1} - \alpha_k| \cdot \|\vec{p}(O)\| < 0.00001$$
$$\alpha = \alpha_0$$

4.2.3 Endbedingung für das Algorithmus

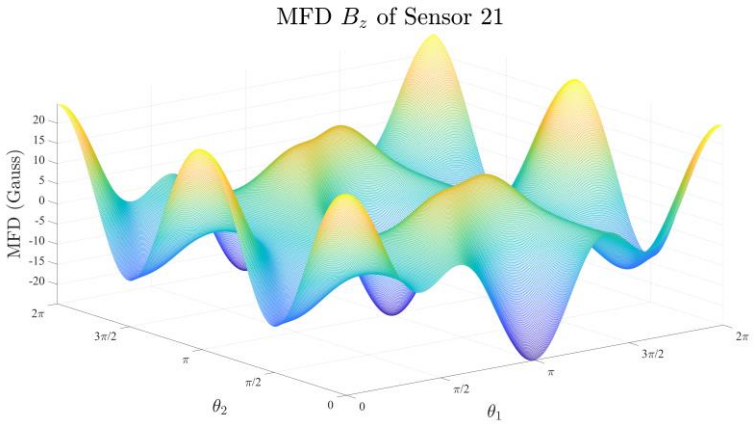
1. Kriterium

$$\left\| G\left(O^{(n)}\right) - G\left(O^{(n-1)}\right) \right\|_2 < 0.0001$$

2. Kriterium

$$n > 100$$

4.2.4 Problem

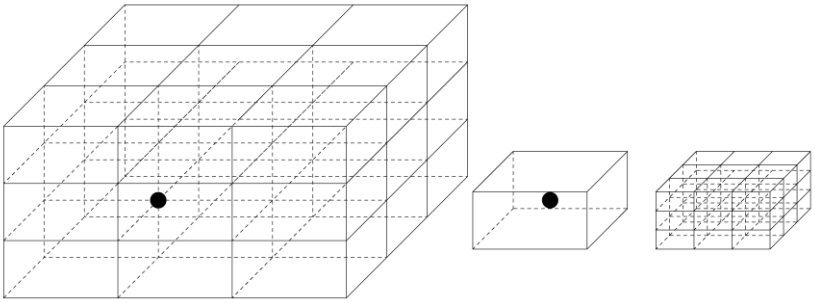


Problem von Gradienten Methode ist, dass es zu viele lokale Minimum gibt. Demzufolge ist es fast unmöglich, globale Minimum zu finden.

4.3 Robuster Lokalisierungsalgorithmus

Als gezeigt, es gibt zu viele lokalen Minimum, demzufolge geht die Gradient Methode nicht. Man muss eine robustere Methode wählen, also z.B. Box-Suchen Methode.

4.3.1 Vorgehen



1. Ich sample gleichmäßig verteilte n Knoten in den Suchraum.
2. Da es insgesamt 5 Dimensionen gibt, werden die MFD an n^5 Knoten gerechnet.
3. Danach wähle ich die Knoten, deren MFD am ähnlichsten wie detektierte Signale ist.
4. Dann verkleinere ich den Suchraum auf $\left(\frac{1}{n-1}\right)^5$.
6. Rekursive berechnen bis zu der Suchraum kleiner als ε mm ist.

In meinem Code wähle ich $n = 4,\ \varepsilon = 0.3$

4.3.2 Ergebnis

Der Algorithmus funktioniert sehr gut, robust und findet immer eine gute Lösung.

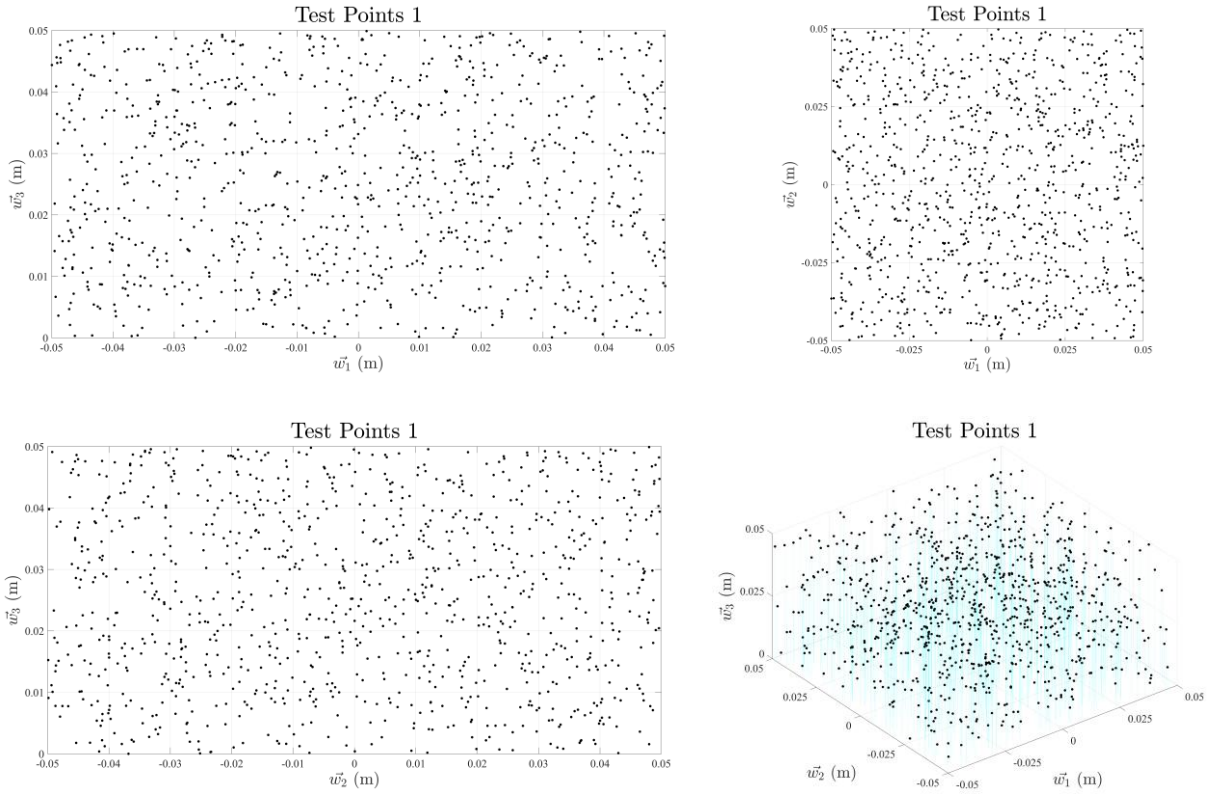
Aber der Nachteil ist, dass der Rechenaufwand hängt stark von der Anzahl der Knoten ab. Bei $n \geq 6$ läuft der Code sehr langsam.

5 Lokalisierungsversuche

5.1 Hardware Bewertung

5.1.1 Test-Punkte

Zufällig Test-Punkte generieren: ich generiere 1000 Testpunkte und deren entsprechenden MFD an jedem Sensor.



5.1.2 Sensordaten

Um die Kenngrößen zu analysieren, bilde ich 4 Gruppe, in jeder Gruppe gibt es eine Kontrollgruppe und 4 Experimentgruppe

ADC Auflösung

INDEX	ADC	Drift	Sensitivität
1	12	2%	5‰
2	14	2%	5‰
0	16	2%	5‰
3	18	2%	5‰
4	20	2%	5‰

Also die ADC Auflösungen sind je nach der Gruppe besser.

Drift

INDEX	ADC	Drift	Sensitivität
5	16	0.50%	5‰
6	16	1%	5‰
0	16	2%	5‰
7	16	4%	5‰
8	16	8%	5‰

Also die Drifte sind je nach der Gruppe schlechter.

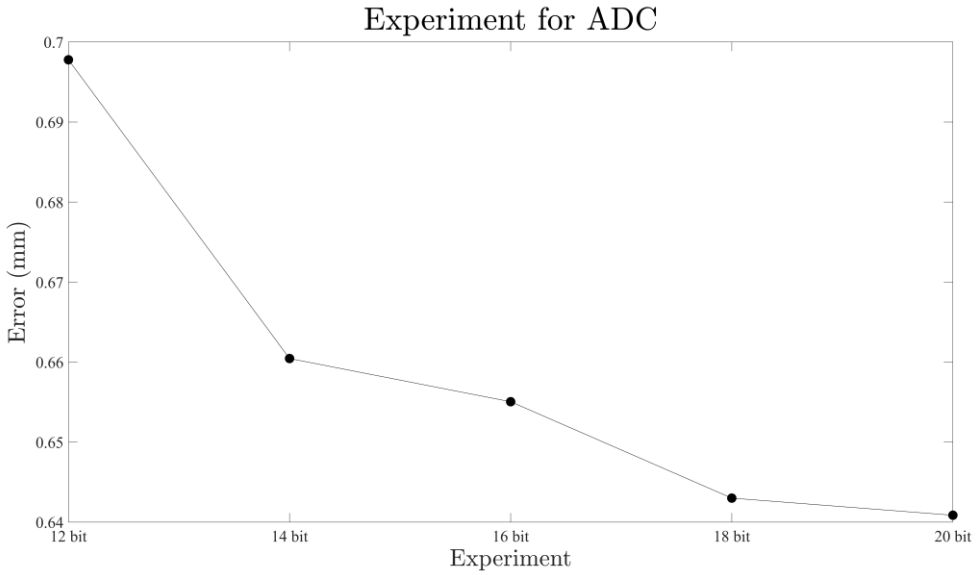
Es ist möglich, dass der Einfluss der Kenngrößen auf die Lokalisierungsfehler abhängig voneinander sind. Z.B., wenn die ADC Auflösung zu schlecht ist (also die Quantisierungsfehler zu groß), ist die Verbesserung des Drifts sinnlos.

Deswegen in folgende Gruppe verbessere ich gleichzeitig alle Kenngrößen.

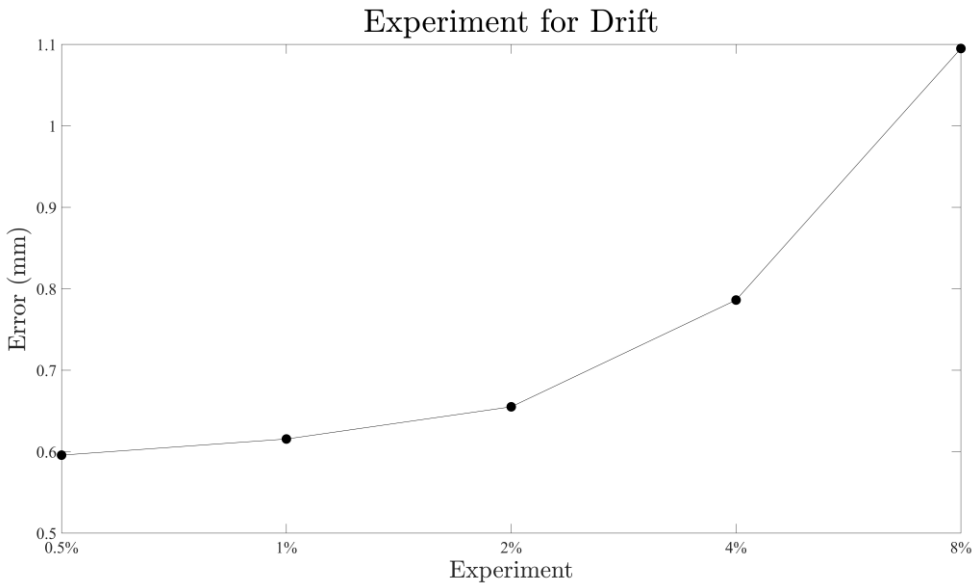
INDEX	ADC	Drift	Sensitivität
0	16	2%	5‰
13	18	1%	5‰
14	20	0.5%	10‰
15	50	0	100‰

5.1.3 Ergebnisse und Analyse

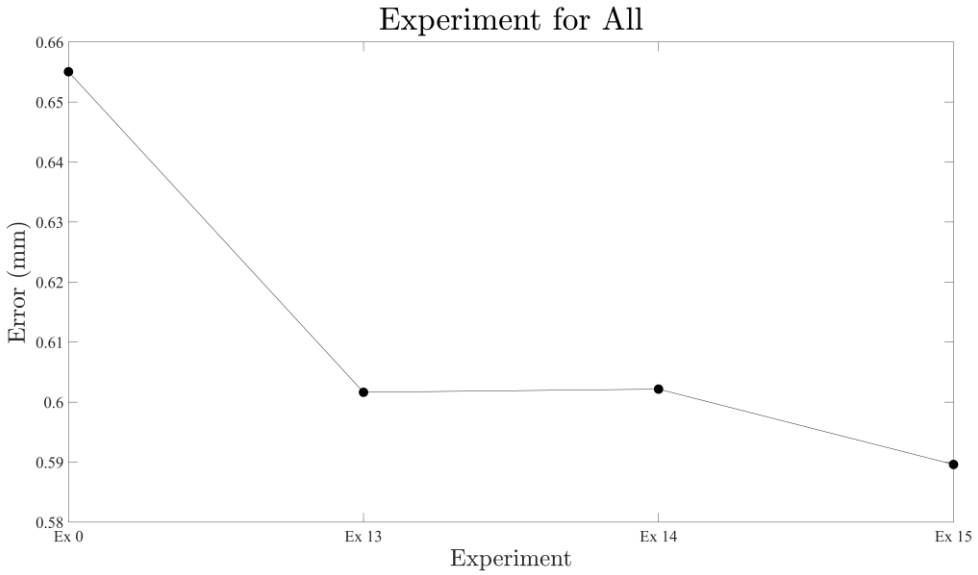
Sensordaten und Fehler



Je größere Auflösung von AD-Wandler ist, desto bessere Lokalisierungsexaktheit es gibt. Aber wenn sie mehr als 18-bit ist, gibt es keine Verbessrung mehr. Weil ab 18-bit, 5% Sensitivität kann Detektierbare-Fläche das Workspace gut decken.

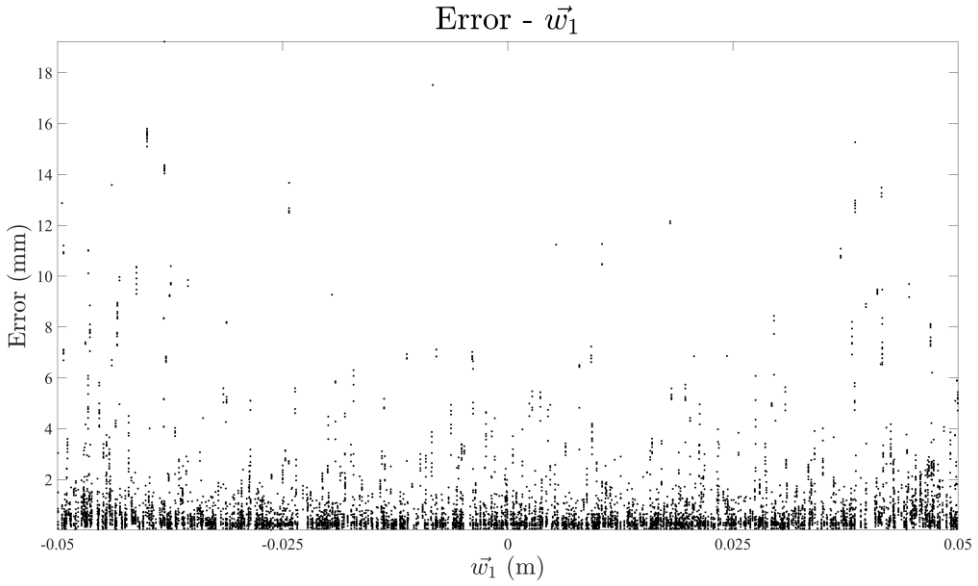


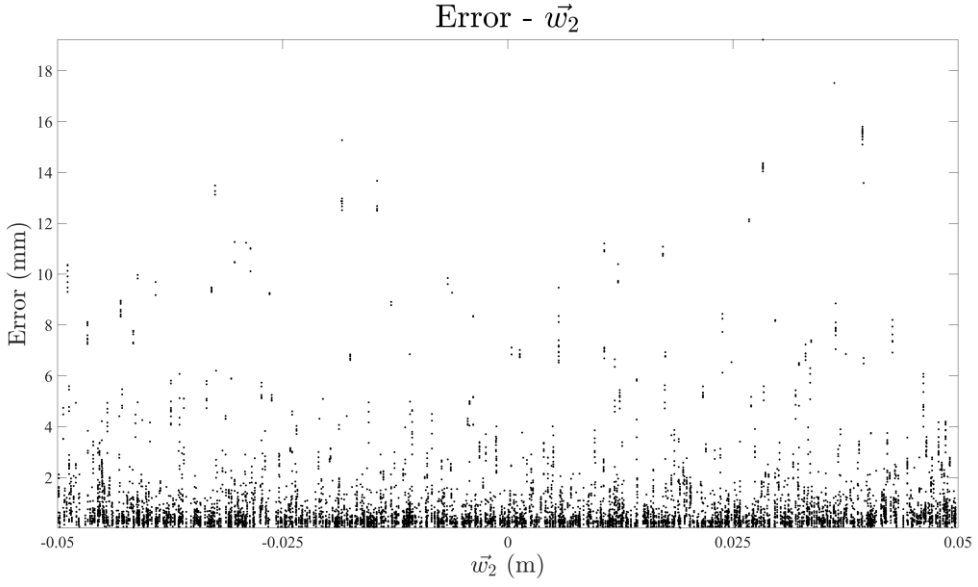
Auffällig ist, der Drift spielt der wichtigsten Rolle. Der Lokalisierungsfehler steigt sehr schnell mit der wachsenden Drift. Deswegen ist es wichtig, die Sensoren mit niedriger Drift zu nutzen. Aber natürlich ist, je niedriger Drift ist, desto teuer ist die Sensoren.

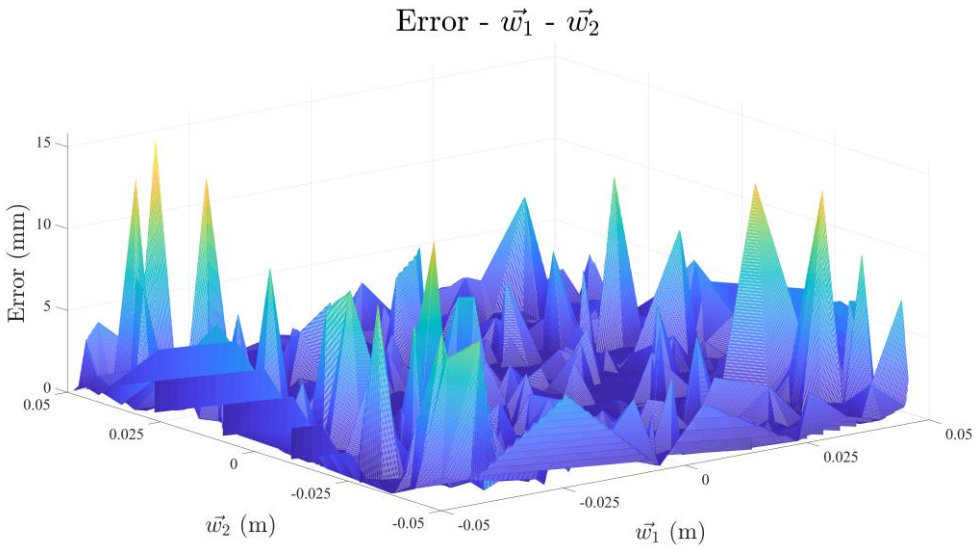


Die Änderung ist eigentlich gleich wie Drift. D.h. Der Drift ist der wichtigste Parameter bei der Wahl des Sensors.

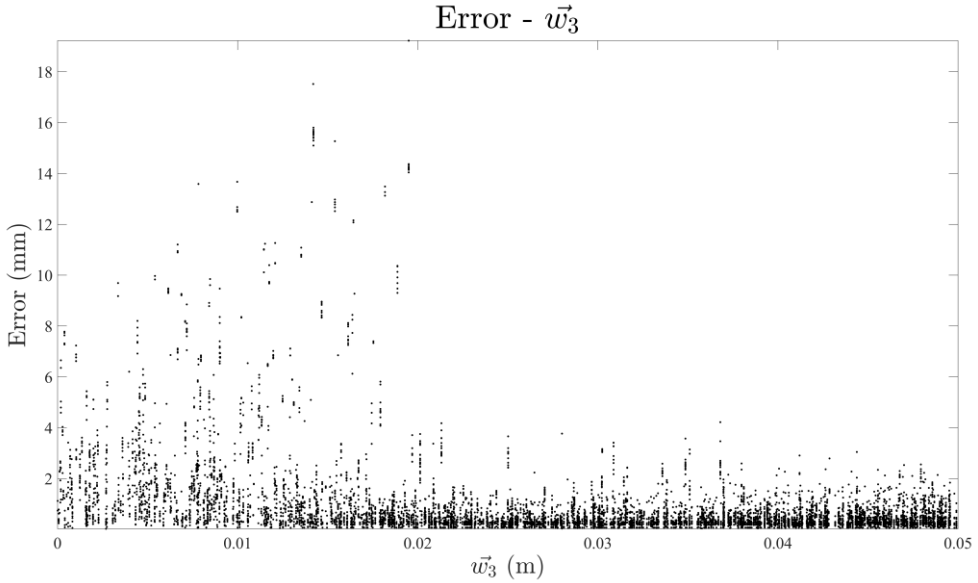
Positionen und Fehler







Aus diesen 3 Bilden kann man nicht sagen, ob die Lokalisierungsfehler von den Positionen des Testpunkts abhängig sind.



Aus dem Bild kann man sagen, dass je großer $\vec{w}_3$ ist, desto bessere Lokalisierung es gibt.

Grund dafür ist, dass die MFD in der Nähe vom Magnet so stark ist, dass die anderen Signale übersehen werden. D.h. die Anzahl der gültigen Sensoren sinkt.

5.2 Workspace Analyse

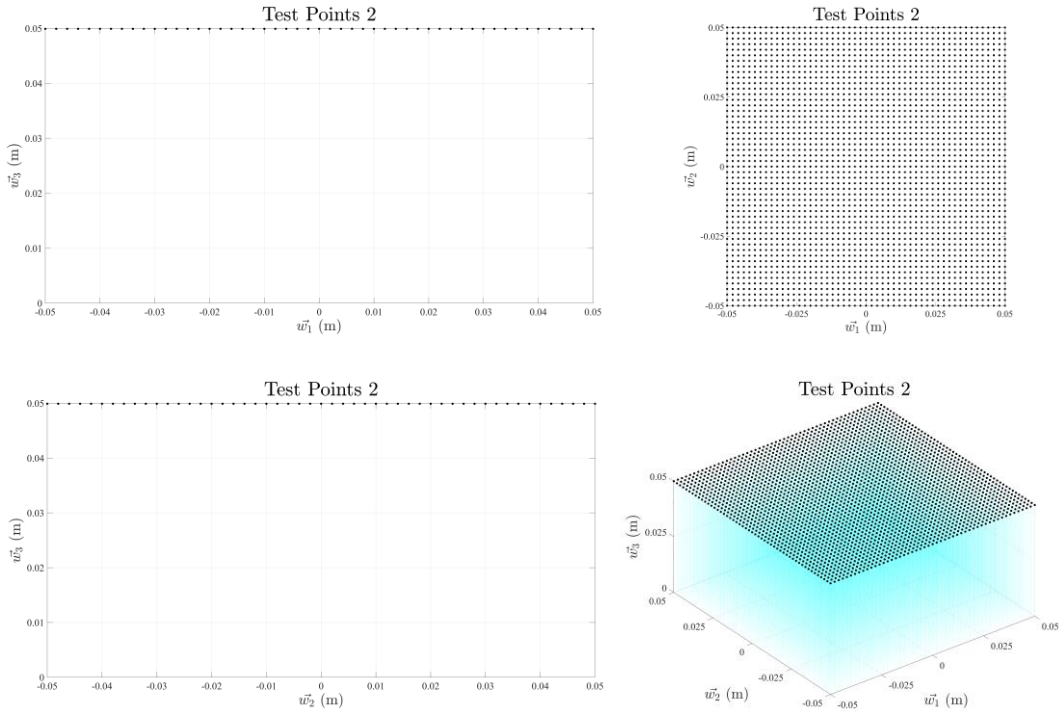
Das Ziel ist, um den Einfluss von w_1 und w_2 auf den Lokalisierungsfehler zu analysieren

5.2.1 Test-Punkte

2. Versuch

Beim 2. Versuch fokussieren wir auf der ganze $\vec{w}_1 - \vec{w}_2$ Ebene mit $\vec{w}_3 = 0.05$. (Weil an $\vec{w}_3 = 0.05$ die Lokalisierungsergebnisse am besten sind.)

Dann sample ich 51×51 gleichmäßig verteilte Test-Punkte. Sieh folgendem Bild. Um das Verhältnis zwischen Positionen und Lokalisierungsfehler zu analysieren.

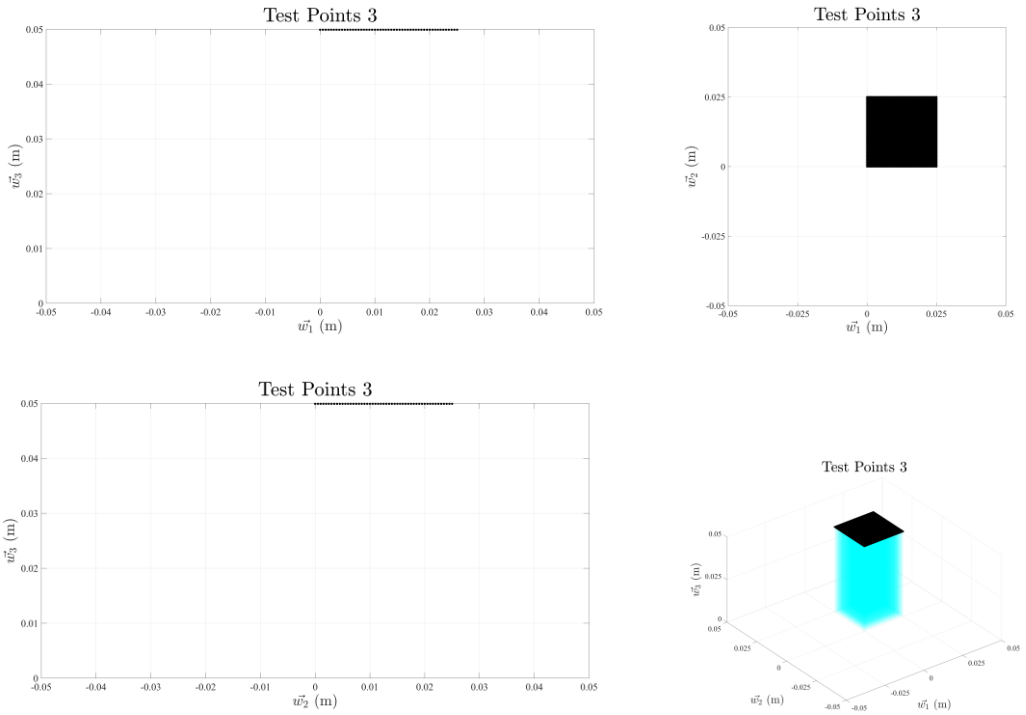


3. Versuch

Beim 3. Versuch fokussieren wir auf ein Segment des Workspace. Also ein Bereich innerhalb 4 Sensoren.

Da der Messaufbau aus mehreren identischen Segmenten, also den Bereichen zwischen 4 Sensoren besteht und wir in den Vorversuchen gesehen haben, dass nur die Sensoren nahe am Magnet eine große Auswirkung auf die Lokalisierung haben, habe ich auch untersucht wie sich der Fehler innerhalb eines Segmentes verhält

Dann sample ich 51×51 gleichmäßig verteilte Test-Punkte. Sieh folgendem Bild. Um das Verhältnis zwischen Positionen und Lokalisierungsfehler zu analysieren.



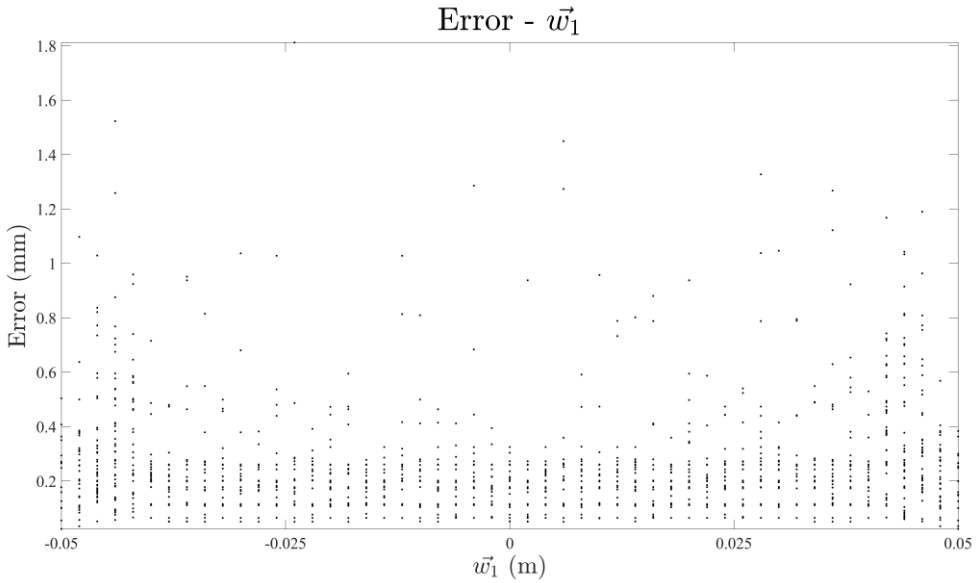
5.2.2 Sensordaten

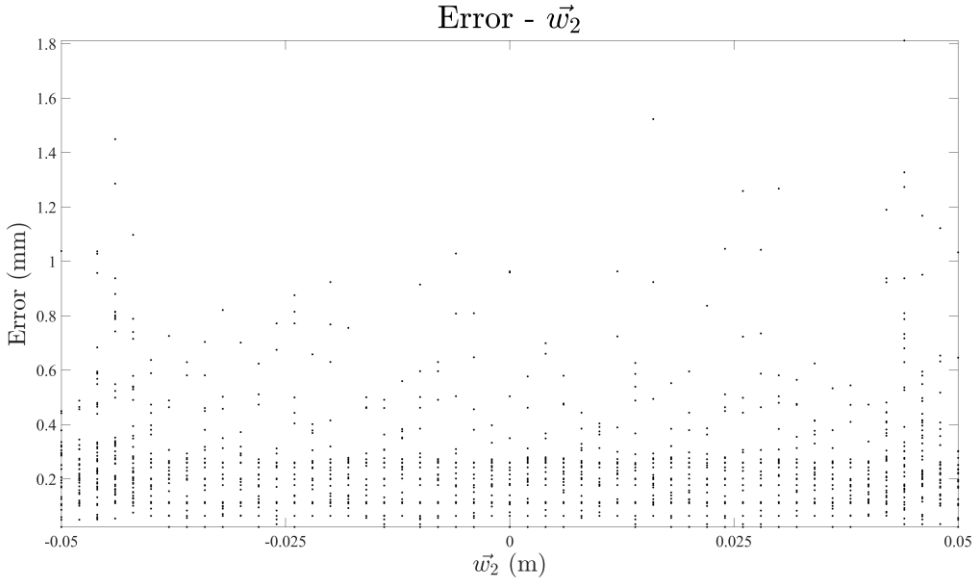
Wählen wir hier die besten (ideale) Sensordaten. Also

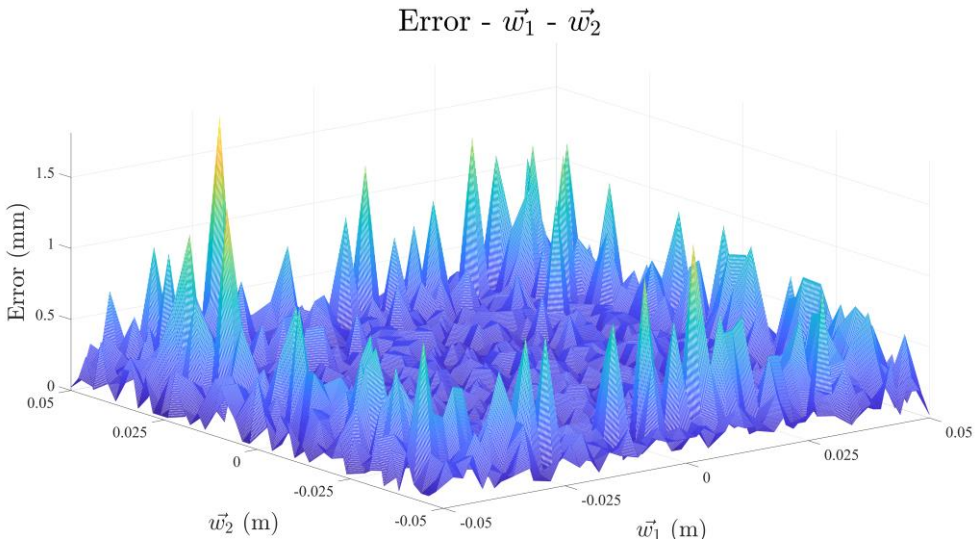
ADC	Drift	Sensitivität
50	0	100‰

5.2.3 Ergebnisse und Analyse

2. Versuch

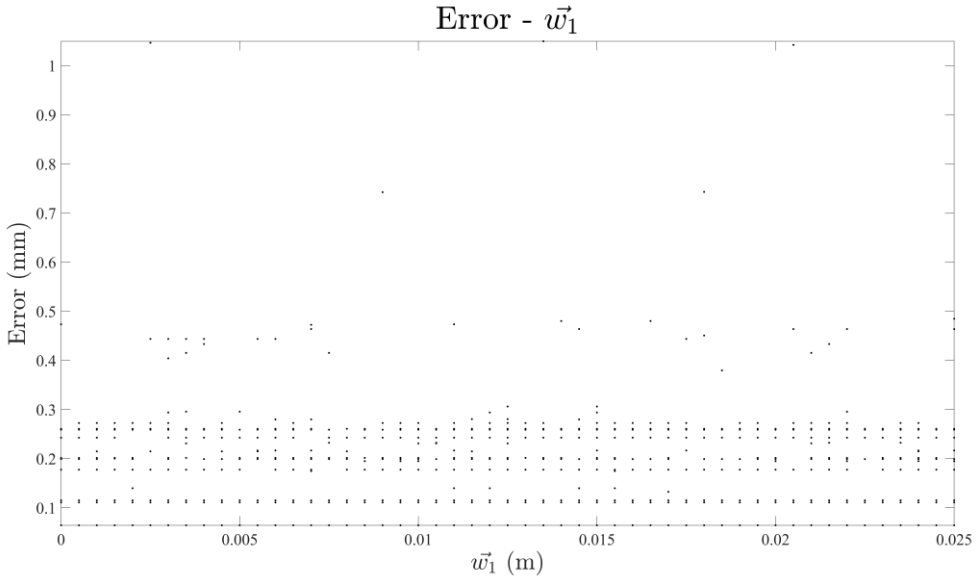


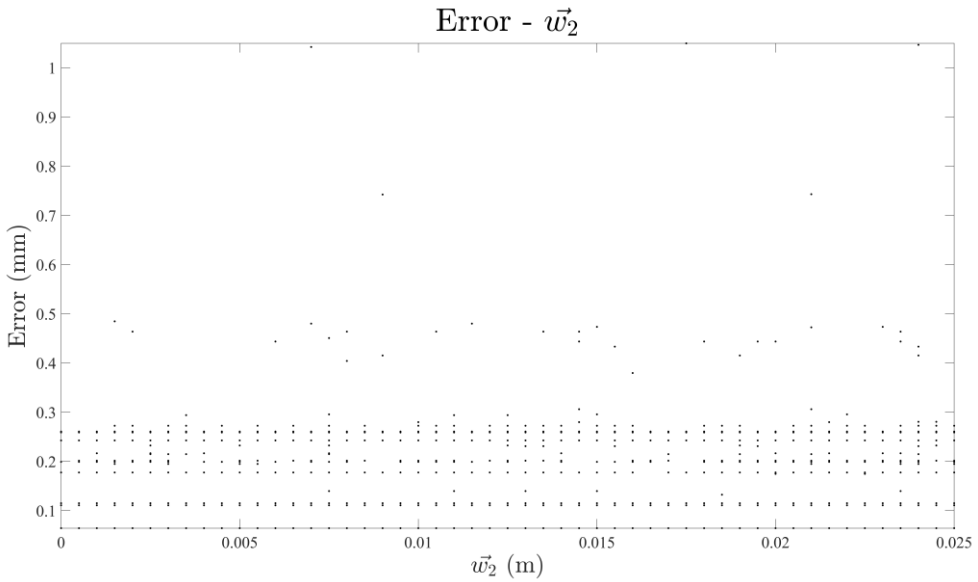


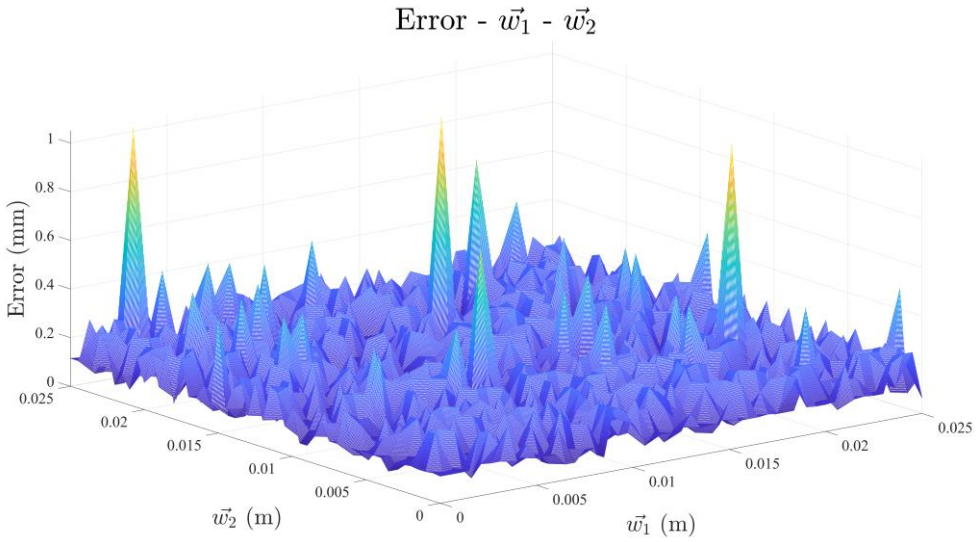


Wir können davon ausgehen, dass die Lokalisierungsfehle ist klein im mittle Teil des Workspace, und groß am Rande.

3. Versuch







Es gibt kein starkes Verhältnis zwischen Positionen und Fehler innerhalb eines Segments.

5.3 Sensorarray Analyse

Das Ziel ist, um den Einfluss der Anzahl der Sensoren auf den Lokalisierungsfehler zu analysieren.

5.3.1 Test-Punkte

Das größte Problem bei Sensorarray ist, dass zu große MFD die Anzahl gültigeren Sensoren senkt. Deswegen müssen wir die Test-Punkte den ganzen Arbeitsraum besitzen. Somit nutze ich zufällige Samplingstrategie, also die gleiche Test-Punkte wie im 1. Versuch.

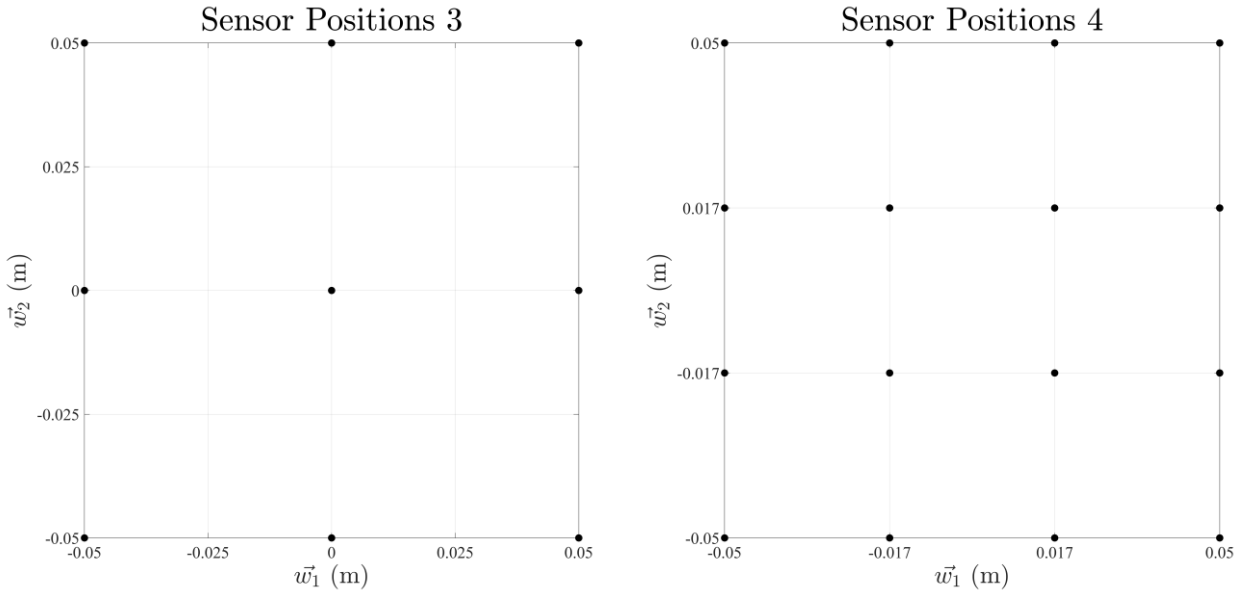
5.3.2 Sensordaten

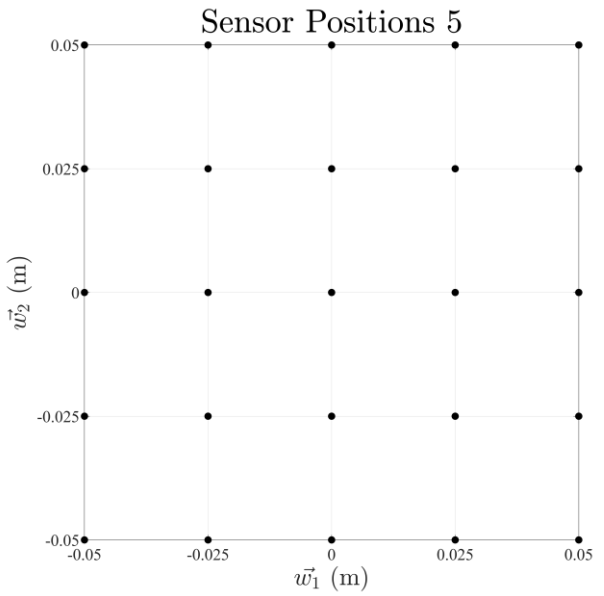
Wählen wir hier die besten (ideale) Sensordaten. Also

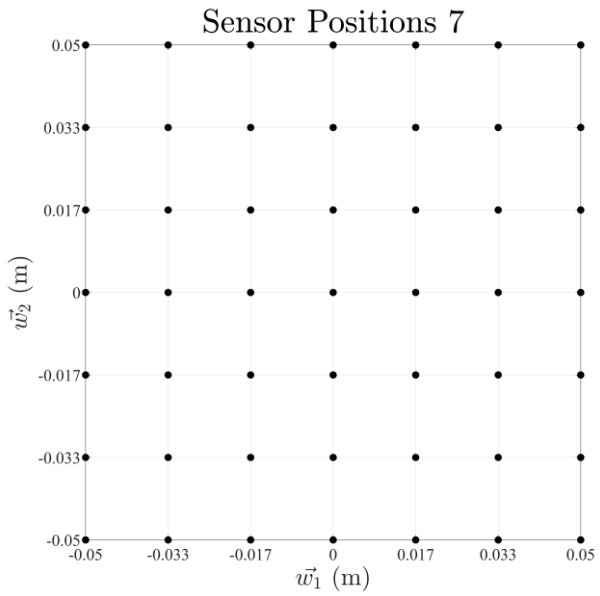
ADC	Drift	Sensitivität
50	0	100‰

5.3.3 Sensorarraydaten

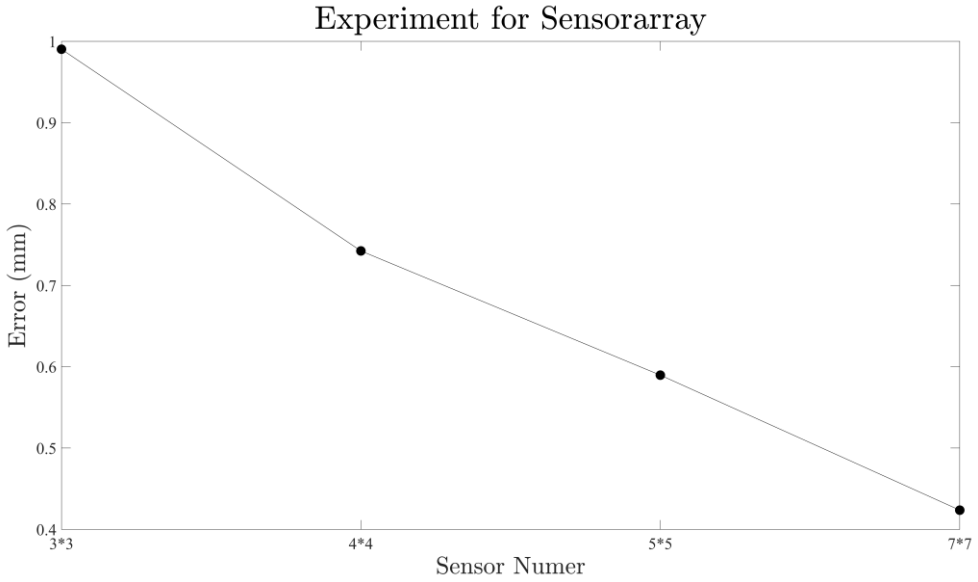
Wir vergleichen hier 4 Sensorarray, nämlich Sensorarray mit 3×3, 4×4, 5×5 und 7×7 gleichmäßig verteilten Sensoren







5.3.4 Ergebnisse und Analyse



Als gezeigt, je mehr Sensoren es in Sensorarray gibt, desto besser das Ergebnis ist.

Z.B wenn der Anzahl der Sensoren von 3 *3 auf 5 *5 erhöhet, senkt der Lokalisierungsfehler um 60%.

Aber betrachtet werden muss, dass der Rechenaufwand und Preis steigt auch sehr stark. Deswegen kann man später auch adaptive Verteilung des Sensors nutzen.

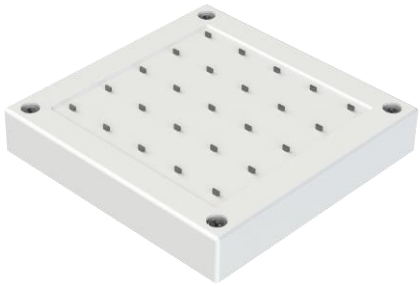
6 Validierung

6.1 Hardware

6.1.1 Hardware

Für die experimentelle Validierung entwarf ich eine Halterung für die gezeigten Sensoren, diesen Magnet und einen Arduino Mega

Sensorarray Halterung: Ich entwerfe eine Halterung und mithilfe 3D-Druckers drucke.



Sensoren

Marke: Honey-Well

Type: analoge Position Sensoren SS94A1

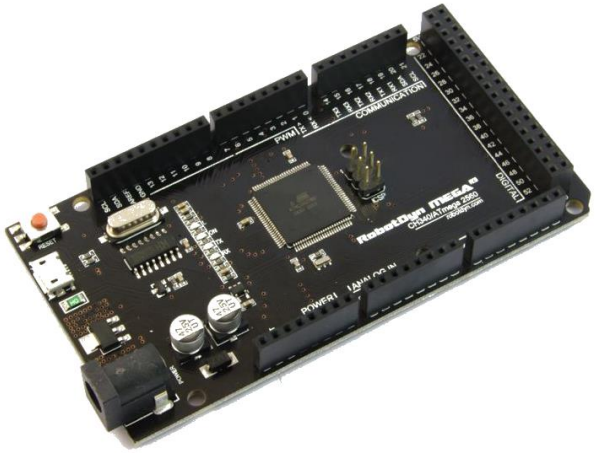
Technische Daten:

Supply Voltage (VDC)	6.6 ~ 12.6
Supply Current (mA)	13 ~ 30
Output Current (mA)	~ 1
Response Time (m sec.)	3
Range (gauss)	-500 ~ +500
Sensitivity (mV/gauss @ 25°C)	5.0 ± 0.1
Vout (0 Gauss @ 25°C)	4.00 ± 0.04
Temperature Error (%/°C)	± 0.02

Arduino Mega

53 Digitale Input

16 Analoge Input mit 10-bit AD Wandler



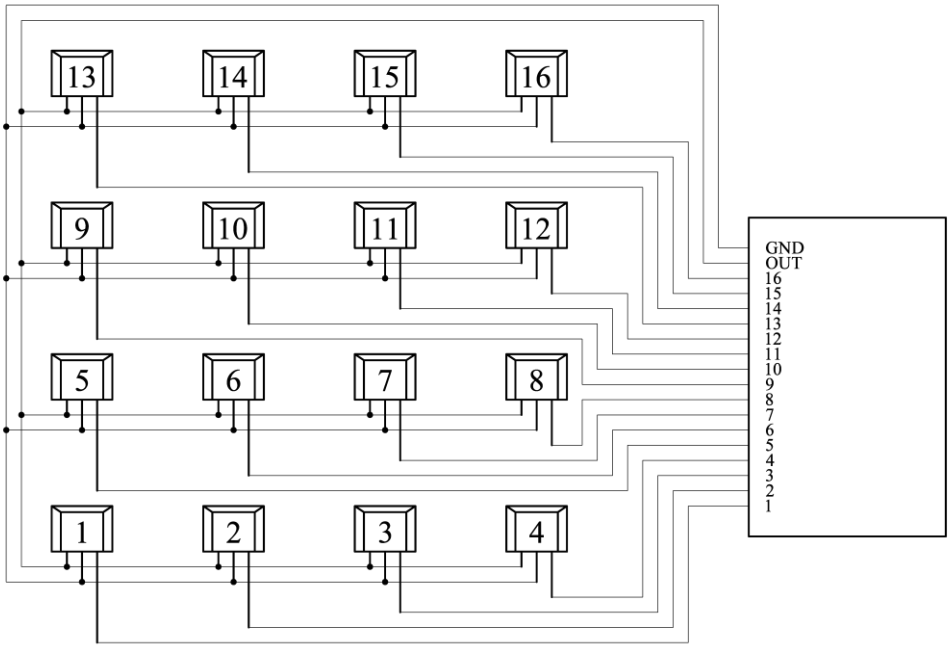
Magnet

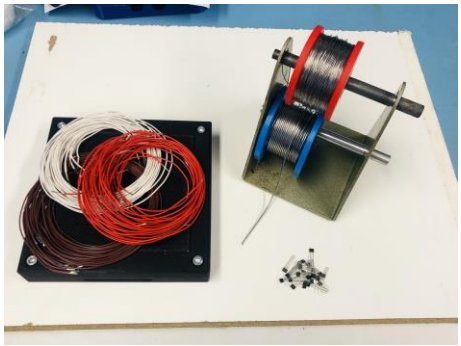
Magnet Größe $R_a = 9$ mm, $L = 5$ mm

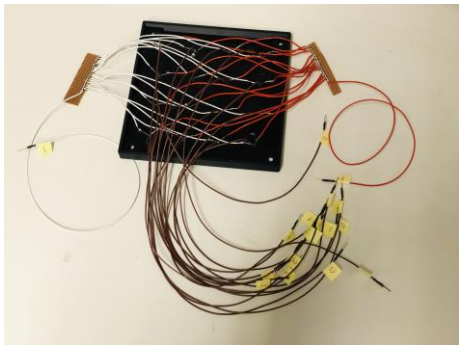
Magnetwerkstoff: Magnetisierungsgrad N52

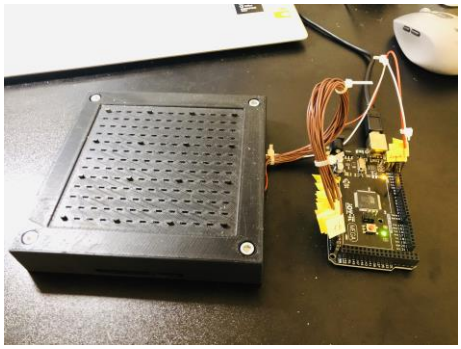
Magnetische Remanenz $B_r = 1480$ mT $=$ 1.48 T

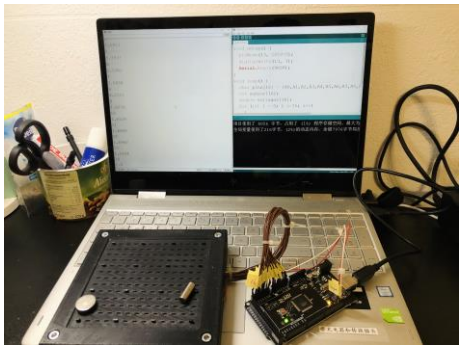
6.1.2 Verkabelung











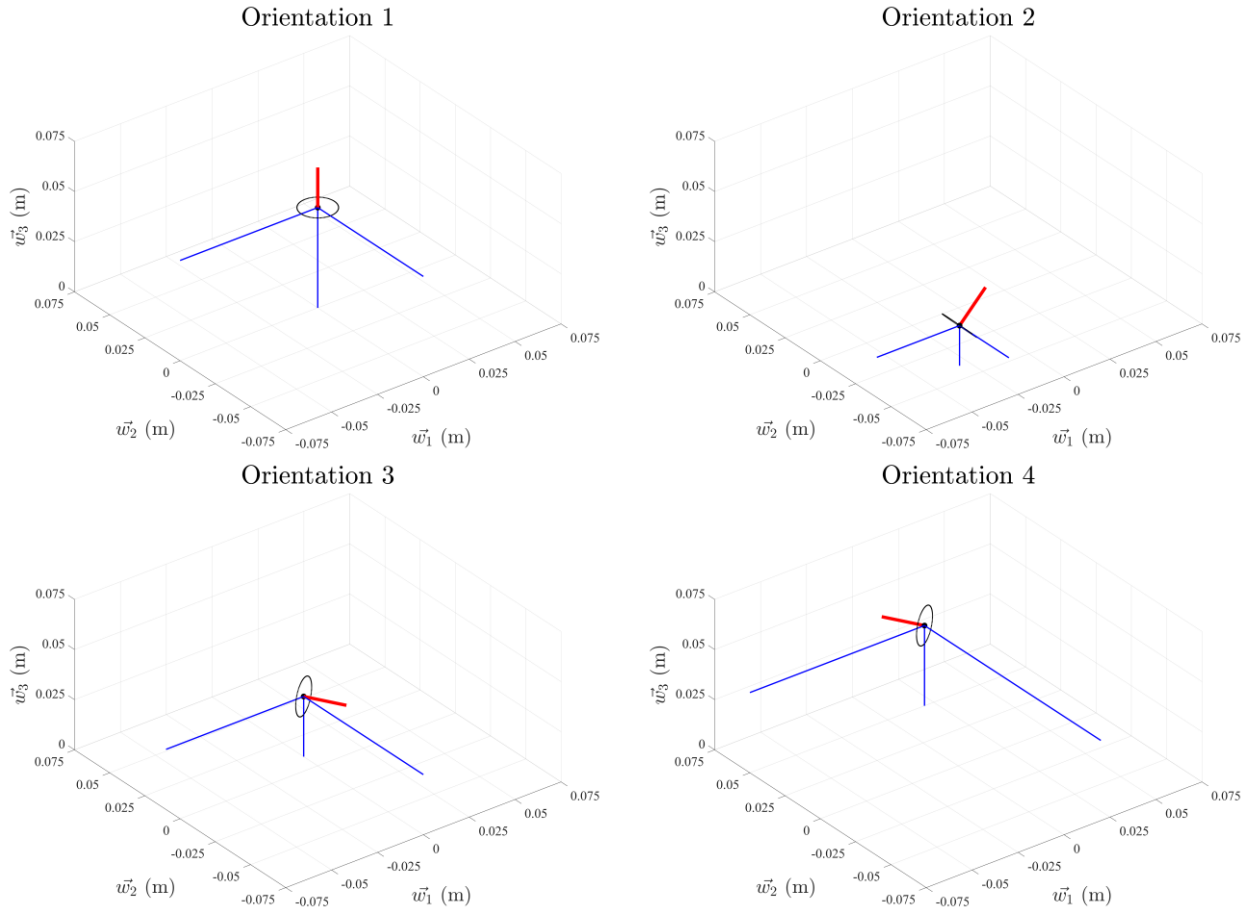
6.2 Vergleichen

Ich habe die Lokalisierung an 4 Positionen getestet. Ich wähle manuelle 4 Orientierungen. Da die Orientierung, besonders die Rotation nicht in Weltkoordinaten aufgeführt, schrieb ich eine MATLAB-Skript um Orientierung anschaulich zu zeigen

Validierungspunkte

Punkte	$\vec{w}_1$ (m)	$\vec{w}_2$ (m)	$\vec{w}_3$ (m)	θ_1	θ_2
1	0	0	0.05	0	0
2	-0.03	-0.04	0.02	0	$\pi/2$
3	0	0.01	0.03	$\pi/2$	$\pi/6$
4	0.02	0.05	0.04	$-\pi/2$	$-\pi/6$

zeichne nun die Orientierung



6.3 Ergebnis

Theorietische Lösung

Sensor Nummer	MFD (Gauß)			
	Punkt 1	Punkt 2	Punkt 3	Punkt 4
1	-2.36	3.5	3.99	-1.61
2	-5.65	3	4.09	-3.39
3	-5.65	1.38	-5.14	-4.38
4	-2.36	0.51	-4.14	9
5	-1.88	11.37	5.92	-1.06
6	-7.11	7.73	11.4	-1.85

7	-7.11	1.63	-37.57	1.7
8	-1.88	0.35	-2.39	24.74
9	1.88	-51.51	0.99	-0.09
10	7.11	-13.58	-12.16	0.87
11	7.11	-0.6	-17.4	5.52
12	1.88	-0.07	10.52	6.29
13	2.36	0	-1.8	0.37
14	5.65	0	-5.9	1.15
15	5.65	0	-1.17	2.29
16	2.36	0	4.92	1.25

Detektierte MFD

Sensor Nummer	Spannung (V)			
	Punkt 1	Punkt 2	Punkt 3	Punkt 4
1	1.6178	1.6178	1.6129	1.6129
2	1.608	1.5836	1.5787	1.5836
3	1.608	1.5982	1.5934	1.608
4	1.6129	1.6031	1.6129	1.6227
5	1.5982	1.6031	1.5934	1.5982
6	1.6373	1.5934	1.5836	1.5982
7	1.6764	1.6129	1.6471	1.6276
8	1.6129	1.6129	1.6373	1.6227
9	1.6227	1.8426	1.6422	1.6325
10	1.5787	1.6373	1.652	1.6178
11	1.5836	1.6129	1.608	1.6031
12	1.608	1.6227	1.608	1.608
13	1.5885	1.4467	1.608	1.5934
14	1.608	1.608	1.6373	1.6129
15	1.6031	1.6276	1.6129	1.6178
16	1.6227	1.6178	1.6178	1.6325

Kalibrierung:

Laut Superpositionsprinzip können wir die Nullausgangsspannung (lässt alle magnetische Materie entfernt vom Sensorarray) einfach von den detektierten Signalen abziehen.

Rechnen wir nun die theoretische Lösung in die Spannung um:

$$V_{theorie} = B_{theorie} \cdot S + V_z$$

$$V_{quantisierung} = \left\lfloor \frac{\left(V_{theorie} + \frac{5}{2048}\right)}{\frac{5}{1024}} \right\rfloor \times \frac{5}{1024}$$

Bewertung:

Nehmen wir L2 Norm als das Güterkriterien um die theoretische Lösung zu bewerten.

$$G = \left\| V_{ist} - V_{quantisierung} \right\|_2$$

für jede Sensor gilt der Fehler

$$G_s = \frac{G}{16}$$

Sensoren	Validierung 1		Validierung 2		Validierung 3		Validierung 4	
	Ist	Theo.	Ist	Theo.	Ist	Theo.	Ist	Theo.
1	1.6178	1.6210	1.6178	1.6113	1.6129	1.6113	1.6129	1.6210
2	1.6080	1.6064	1.5836	1.5917	1.5787	1.5917	1.5836	1.6015
3	1.6080	1.6015	1.5982	1.5917	1.5934	1.6015	1.608	1.6015
4	1.6129	1.6113	1.6031	1.6064	1.6129	1.6162	1.6227	1.5917
5	1.5982	1.5917	1.6031	1.5673	1.5934	1.5771	1.5982	1.5917
6	1.6373	1.6162	1.5934	1.5917	1.5836	1.5820	1.5982	1.6064
7	1.6764	1.6308	1.6129	1.6162	1.6471	1.6796	1.6276	1.6162
8	1.6129	1.6259	1.6129	1.6210	1.6373	1.6259	1.6227	1.5820
9	1.6227	1.6259	1.8426	1.7138	1.6422	1.6259	1.6325	1.6259
10	1.5787	1.6015	1.6373	1.6357	1.6520	1.6308	1.6178	1.6113
11	1.5836	1.6162	1.6129	1.6308	1.6080	1.6552	1.6031	1.6162
12	1.6080	1.6259	1.6227	1.6259	1.6080	1.6113	1.608	1.6162
13	1.5885	1.6015	1.4467	1.6015	1.6080	1.6064	1.5934	1.6015
14	1.6080	1.6113	1.6080	1.6210	1.6373	1.6308	1.6129	1.6210
15	1.6031	1.6064	1.6276	1.6162	1.6129	1.6210	1.6178	1.6162
16	1.6227	1.6308	1.6178	1.6308	1.6178	1.6259	1.6325	1.6308
Fehler (V)	0.004405		0.012942		0.004349		0.003823	

Diskussion:

Da der Quantisierungsstufe

$$\Delta V = \frac{5}{1024} \approx 0.005\,V = 5\text{ mV}$$

ist, sind die Fehler von jedem Sensor bei 4 Validierungspunkten innerhalb einer Quantisierungsstufe. Der Fehler besteht hauptsächlich aus 3 Teilen

Drift des Sensors;

nicht genügende Versorgungsspannung wegen der fehlenden elektrischen Bauelementen;

Positionsungenauigkeit, denn alle 4 Orientierungen wurden manuell gehalten.

Deswegen halte ich dafür, dass aus diesen Ergebnissen kann man schließen dass die Theorie valide ist.

7 Diskussion

7.1 Meine Arbeit

Theoretischer Teil

Bei der Magnet-Modellierung habe ich

Theoretisch hergeleitet;

Numerisch gelöst;

Koordinaten umgerechnet;

Hardware validiert.

Bei der Bildung des Workspaces habe ich

Welt- und Magnetkoordinaten gebildet und umgerechnet;

Parameter der Sensoren analysiert;

Sensorarray entwürfe.

Bei der Lokalisierungsphase habe ich

zwei Lokalisierungsalgorithmen entwickelt.

Bei der Lokalisierungsversuche habe ich

Parameter der Sensoren analysiert;

Workspace analysiert;

Sensorarray analysiert.

Software Realisierung: mehr als 1000 Zeile Matlab Codes

Hardware Realisierung: Sensoren, Halterung, Arduino, Löten, Verkabelung, mit PC Verbindung

7.2 Verbesserung

1. Ausführungszeit des Codes

Optimierung der Lokalisierungsalgorithmen, neue Algorithmen zu entwickeln;

Optimierung des Codes: Loops zu vermeiden, Parameterübergabe zu vermeiden;

Schnellere Sprach: C++.

2. Genauigkeit der Lokalisierung

Erhöhe die Genauigkeit der Algorithmen im Code. (Mehr Rechenaufwand);

Trajektorie Analyse (Mehr Rechenaufwand);

Adaptive Auflösung des Sensorarrays (Modellbasiert);

Informationsfusion (fusioniere Regelungsinformation, kinematisches Modell des Softroboters usw. mithilfe von z.B. Kalman-Filters).

7.3 Ausblick

1. Verwendung der kodierten Magneten
2. mehrere Magnete Lokalisierung
3. Tracking nur Positionen (viel schneller und verbreitetere Anwendungen, z.B. die Lokalisierung des Mikroroboters in Menschkörper)

8 Wichtige Codes

8.1 Project.m

```
clc
clear
% Decorator. 1 is activited, 0 is deactivited
decroator = 1;
% Plot and Mesh. 1 is activited, 0 is deactivited, 2 means to output the figure as png-file
figureplot = 0;

% Workspace
%Size of the Workspaces in Meter m
W = [0.1,0.1,0.05];

% Magnet
% Radius of the Magnet in Meter m
Magnet_radius = 0.009;
% Altitude of the Magnet in Meter m
Magnet_altitude = 0.005;
% Magnetic Remanenz in T
Magnet_remanenz = 1.48;
% Number of the Nodes
Node_Number = 500;
% Width and Length of the calculated area
Area_Width = 0.15;
Area_Length = 2 * Area_Width;
mu0 = 4*pi*10^-7;

% Hardwares
% Drift, Zerovoltage, Range, Sensitivity
Sensor_data = [2, 4, 500, 5/1000];
% number of sensors on each edge
Sensor_number = 5;
ADC_bit = 16;
% AD Converter bit
% test_data for sensors
Test_Data = [2, 4, 500, 5/1000, 16;...
             2, 4, 500, 5/1000, 12;...
             2, 4, 500, 5/1000, 14;...
             2, 4, 500, 5/1000, 18;...
             2, 4, 500, 5/1000, 20;...
             0.5, 4, 500, 5/1000, 16;...
             1, 4, 500, 5/1000, 16;...
             4, 4, 500, 5/1000, 16;...
             8, 4, 500, 5/1000, 16;...
             2, 4, 500, 1/1000, 16;...
             2, 4, 500, 2/1000, 16;...
             2, 4, 500, 10/1000, 16;...
             2, 4, 500, 20/1000, 20;...
             1, 4, 500, 5/1000, 18;...
             0.5, 4, 500, 10/1000, 20;...
             0, 4, 500, 100/1000, 50];
% data(1:4)=Sensordata, data(5) = k (ADC bit)

% Movement of Magnet
% maximal velocity of magnet
max_movespeed = 1;
% maximal rotation speed of magnet
max_rotatespeed = pi;

% Calculate the Magnetic Field Distribution
% Decroator('Calculating Magetic Field Strength ...',decroator);
% [MFS_y,MFS_z,Coordinate_q2,Coordinate_q3] =
Magnetfield(Magnet_radius,Magnet_altitude,Magnet_remanenz,Area_Width,Area_Length,Node_Number,'main',1);
% last parameter 1 = write excel file, 0 = do not write

% reading MFS
Decroator('Reading Magnetic Field Strength ...',decroator);
MFS_y = xlsread('Hy_main.xlsx');
MFS_z = xlsread('Hz_main.xlsx');
Coordinate_q2 = xlsread('q2_main.xlsx');
Coordinate_q3 = xlsread('q3_main.xlsx');
MFS_y_show_flat = xlsread('Hy_show_flat.xlsx');
MFS_z_show_flat = xlsread('Hz_show_flat.xlsx');
MFS_y_show_long = xlsread('Hy_show_long.xlsx');
MFS_z_show_long = xlsread('Hz_show_long.xlsx');
Coordinate_q2_show = xlsread('q2_show.xlsx');
Coordinate_q3_show = xlsread('q3_show.xlsx');

MFS_value = sqrt(MFS_y.^2+MFS_z.^2);
```

```
MFS_value_show_flat = sqrt(MFS_y_show_flat.^2+MFS_z_show_flat.^2);
MFS_value_show_long = sqrt(MFS_y_show_long.^2+MFS_z_show_long.^2);

% MFS into MFD in Gauss
Decroator('Converting MFS into Magnetic Flux Density ...',decroator);
MFD_y = MFS_y * mu0 * 10000;
MFD_z = MFS_z * mu0 * 10000;
MFD_value = MFS_value * mu0 * 10000;
MFD_y_show_flat = MFS_y_show_flat * mu0 * 10000;
MFD_z_show_flat = MFS_z_show_flat * mu0 * 10000;
MFD_value_show_flat = MFS_value_show_flat * mu0 * 10000;
MFD_y_show_long = MFS_y_show_long * mu0 * 10000;
MFD_z_show_long = MFS_z_show_long * mu0 * 10000;
MFD_value_show_long = MFS_value_show_long * mu0 * 10000;
mesh3D(Coordinate_q2,Coordinate_q3,MFD_y,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD of the Magnet in
$\vec{i}_{2}$ Direction','MFD_i2',figureplot);
mesh3D(Coordinate_q2,Coordinate_q3,MFD_z,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD of the Magnet in
$\vec{i}_{3}$ Direction','MFD_i3',figureplot);
mesh3D(Coordinate_q2,Coordinate_q3,MFD_value,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD of the
Magnet','MFD',figureplot)
mesh3D(Coordinate_q2_show,Coordinate_q3_show,MFD_y_show_flat,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD of
the flat Magnet in $\vec{i}_{2}$ Direction','MFD_i2_flat',figureplot);
mesh3D(Coordinate_q2_show,Coordinate_q3_show,MFD_z_show_flat,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD of
the flat Magnet in $\vec{i}_{3}$ Direction','MFD_i3_flat',figureplot);
mesh3D(Coordinate_q2_show,Coordinate_q3_show,MFD_value_show_flat,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD
of the flat Magnet','MFD_flat',figureplot)
mesh3D(Coordinate_q2_show,Coordinate_q3_show,MFD_y_show_long,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD of
the long Magnet in $\vec{i}_{2}$ Direction','MFD_i2_long',figureplot);
mesh3D(Coordinate_q2_show,Coordinate_q3_show,MFD_z_show_long,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD of
the long Magnet in $\vec{i}_{3}$ Direction','MFD_i3_long',figureplot);
mesh3D(Coordinate_q2_show,Coordinate_q3_show,MFD_value_show_long,'$q_{2}$ (m)','$q_{3}$ (m)','MFD (Gauss)','MFD
of the long Magnet','MFD_long',figureplot)

% Grandient
Decroator('Calculating Gradient of MFD ...',decroator);
MFD_Gradient_value = grad(MFD_value,Coordinate_q2,Coordinate_q3);
MFD_Gradient_y = grad(MFD_y,Coordinate_q2,Coordinate_q3);
MFD_Gradient_z = grad(MFD_z,Coordinate_q2,Coordinate_q3);
mesh3D(Coordinate_q2(1:end-1),Coordinate_q3(1:end-
1),MFD_Gradient_value(:,:,1),'$q_{2}$ (m)','$q_{3}$ (m)','$G_{m,y}$ (Gauss/m)','Gradient
$\frac{\partial{B_{m}}}{\partial{q_{2}}}$','Gradient_m_i2',figureplot);
mesh3D(Coordinate_q2(1:end-1),Coordinate_q3(1:end-
1),MFD_Gradient_value(:,:,2),'$q_{2}$ (m)','$q_{3}$ (m)','$G_{m,z}$ (Gauss/m)','Gradient
$\frac{\partial{B_{m}}}{\partial{q_{3}}}$','Gradient_m_i3',figureplot);
mesh3D(Coordinate_q2(1:end-1),Coordinate_q3(1:end-
1),MFD_Gradient_y(:,:,1),'$q_{2}$ (m)','$q_{3}$ (m)','$G_{y,y}$ (Gauss/m)','Gradient
$\frac{\partial{B_{y}}}{\partial{q_{2}}}$','Gradient_y_i2',figureplot);
mesh3D(Coordinate_q2(1:end-1),Coordinate_q3(1:end-
1),MFD_Gradient_y(:,:,2),'$q_{2}$ (m)','$q_{3}$ (m)','$G_{y,z}$ (Gauss/m)','Gradient
$\frac{\partial{B_{y}}}{\partial{q_{3}}}$','Gradient_y_i3',figureplot);
mesh3D(Coordinate_q2(1:end-1),Coordinate_q3(1:end-
1),MFD_Gradient_z(:,:,1),'$q_{2}$ (m)','$q_{3}$ (m)','$G_{z,y}$ (Gauss/m)','Gradient
$\frac{\partial{B_{z}}}{\partial{q_{2}}}$','Gradient_z_i2',figureplot);
mesh3D(Coordinate_q2(1:end-1),Coordinate_q3(1:end-
1),MFD_Gradient_z(:,:,2),'$q_{2}$ (m)','$q_{3}$ (m)','$G_{z,z}$ (Gauss/m)','Gradient
$\frac{\partial{B_{z}}}{\partial{q_{3}}}$','Gradient_z_i3',figureplot);

% Sensor positions in global-coordinaten
Decroator('Calculating Sensor Positions ...',decroator);
Sensorposition_w = sensorposition(W(1),Sensor_number);
plotsensorposition(Sensorposition_w(1,:),Sensorposition_w(2,:),'$\vec{w_{1}}$ (m)','$\vec{w_{2}}$ (m)','Sensor
Positions 5','Sensor Positions 5',figureplot);

% Detected Signal at Sensors
% Magnet Orientation
Orientation = [-0.049,-0.049,0.049,0,0];
test_node = 360;
MFD_at_Sensor = zeros(Sensor_number^2,3,test_node,test_node);
THETA1 = 0:1/test_node*2*pi:2*pi-1/test_node*2*pi;
THETA2 = THETA1;
% s is sensor index
for s = 1:Sensor_number^2
    Decroator(['Calculating Detected Signal at Sensor ',num2str(s),' ...'],decroator);
    for i = 1:test_node
        for j = 1:test_node
            theta1 = (i-1)/test_node*2*pi;
            theta2 = (j-1)/test_node*2*pi;
            Orientation(4:5) = [theta1 theta2];
            [r_cs, theta_k]= coordinatew2i(Orientation,Sensorposition_w(:,s));
            [MFD_y_Vglpunkt,MFD_z_Vglpunkt] =
```

```
itplt(r_cs(2,2),r_cs(2,3),Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number);
           [MFD_at_Sensor_x, MFD_at_Sensor_y, MFD_at_Sensor_z] =
coordinatei2w(Orientation(3),Orientation(4),theta_k,MFD_y_Vglpunkt,MFD_z_Vglpunkt);
           MFD_at_Sensor(s,1,i,j) = MFD_at_Sensor_x;
           MFD_at_Sensor(s,2,i,j) = MFD_at_Sensor_y;
           MFD_at_Sensor(s,3,i,j) = MFD_at_Sensor_z;
       end
    end
    direction = ['x','y','z'];
    for p = 1 : 3
        TITLE = ['MFD $B_{',direction(p),'}$ of Sensor ',num2str(s)];
        filename = ['MFD_B_',num2str(s),'_',direction(p)];
        mesh3Ddetectedsignal(THETA1,THETA2,squeeze(MFD_at_Sensor(s,p,:,:)),'$\theta_{1}$','$\theta_{2}$','MFD
(Gauss)',TITLE,filename,figureplot);
    end
end

% Detectable Area
Decroator('Calculating Detectable Area under different Sensor Data ...',decroator);

% out of range area
D_out_of_range = out_of_range(Coordinate_q2,Coordinate_q3,Sensor_data,MFD_value);
out_of_range_points = area2points(Coordinate_q2,Coordinate_q3,D_out_of_range);
plotarea(out_of_range_points(:,1),out_of_range_points(:,2),'$q_{2}$ (m)','$q_{3}$ (m)','Out of Range','Out of
Range',figureplot);

% detectable area
% ADC bit
ADC_BIT = [12,14,16,18,20];
% Sensitivity
SENSITIVITY = [5/1000, 10/1000, 20/1000];
Detectable_area = NaN(length(ADC_BIT),length(SENSITIVITY),length(Coordinate_q2),length(Coordinate_q3));
Detectable_movement = NaN(length(ADC_BIT),length(SENSITIVITY),length(Coordinate_q2),length(Coordinate_q3));
for i = 1:length(ADC_BIT)
    for j = 1 : length(SENSITIVITY)
        Detectable_area(i,j,:,:) =
detectablearea(Coordinate_q2,Coordinate_q3,D_out_of_range,ADC_BIT(i),SENSITIVITY(j),MFD_Gradient_value);
       Detectable_movement(i,j,:,:) =
detectmovement(Coordinate_q2,Coordinate_q3,SENSITIVITY(j),ADC_BIT(i),MFD_Gradient_value);
       Detectable_points = area2points(Coordinate_q2(1:end-1),Coordinate_q3(2:end-
1),squeeze(Detectable_area(i,j,1:end-1,2:end-1)));
       name1 = {['Detectable Area with ',num2str(ADC_BIT(i)),' bit'];['ADC and
',num2str(SENSITIVITY(j)*1000),'¡ë Sensitivity']};
       filename1 = ['Detectable Area ',num2str(ADC_BIT(i)),' bit ',num2str(SENSITIVITY(j)*1000)];

plotarea(Detectable_points(:,1),Detectable_points(:,2),'$q_{2}$ (m)','$q_{3}$ (m)',name1,filename1,figureplot);
       name2 = {['Detectable Movement with ',num2str(ADC_BIT(i)),' bit'];['ADC and
',num2str(SENSITIVITY(j)*1000),'¡ë Sensitivity']};
       filename2 = ['Detectable Movement ',num2str(ADC_BIT(i)),' bit ',num2str(SENSITIVITY(j)*1000)];
       mesh3Darea(Coordinate_q2(1:end-1),Coordinate_q3(2:end-1),squeeze(Detectable_movement(i,j,1:end-1,2:end-
1)),'$q_{2}$ (m)','$q_{3}$ (m)','Movement (mm)',name2,filename2,figureplot);
    end
end

% Localization Process

% 1st experiment
% Experimenttimes = 1000;
% Decroator('Generating Random Test Points ...',decroator)
% % test points
% Orientation_Given = zeros(Experimenttimes,5);
% % MFD of test points at every sensor
% MFD_Given = zeros(25,Experimenttimes);
% for i = 1 : Experimenttimes
%     [Orientation_given, MFD_given] =
randpoint(Sensorposition_w,Sensor_number,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Num
ber);
%     Orientation_Given(i,:) = Orientation_given;
%     MFD_Given(:,i) = MFD_given;
% end
% xlswrite('testpoints.xlsx',[Orientation_Given,MFD_Given']);

Decroator('Reading Random Test Points ....',decroator);
Test_Points = xlsread('testpoints.xlsx');
Orientation_Given = Test_Points(:,1:5);
MFD_Given = Test_Points(:,6:30)';
Experimenttimes = size(Orientation_Given,1);

plottestpoints(Orientation_Given(:,1),Orientation_Given(:,2),Orientation_Given(:,3),'$\vec{w_{1}}$ (m)','$\vec{w
_{2}}$ (m)','$\vec{w_{3}}$ (m)','Test Points 1','Test_Points_1',figureplot);
```

```
% do experiments under different sensordaten and write them in excel-tables
% for i = 1 : size(Test_Data,1)
%     data = Test_Data(i,:);
%     name = ['Ex_',num2str(i-1)];
%     experiment(name, data, Experimenttimes,
Sensorposition_w,Sensor_number,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number,max_mo
vespeed,max_rotatespeed,Orientation_Given,MFD_Given);
% end

% analyse the results
Decroator('Analysing Results ....',decroator);
exresult = zeros(size(Test_Data,1),Experimenttimes,12);
Error = zeros(Experimenttimes,size(Test_Data,1));
Errorinner5mm = zeros(Experimenttimes,size(Test_Data,1));
for i = 1 : size(Test_Data,1)
    name = ['Ex_',num2str(i-1),'.xlsx'];
    readdata = xlsread(name);
    exresult(i,:,:) = readdata(2:end,:);
    Error(:,i) = exresult(i,:,11)';
end
inner5mm = zeros(size(Test_Data,1),1);
meanerror = zeros(size(Test_Data,1),1);
for i = 1 : size(Test_Data,1)
    for j = 1 : Experimenttimes
        if Error(j,i) <= 5
            inner5mm(i) = inner5mm(i) + 1;
            Errorinner5mm(j,i) = Error(j,i);
        else
            Errorinner5mm(j,i) = NaN;
        end
    end
    meanerror(i) = nanmean(Errorinner5mm(:,i));
end
xlswrite('analysis.xlsx',[inner5mm,meanerror]);

plotresult(meanerror,inner5mm,figureplot)

combineresult = squeeze(exresult(1,:,:));
combineerror = Error(:,1);
for i = 2 : size(Test_Data,1)
    combineresult = [combineresult;squeeze(exresult(i,:,:))];
    combineerror = [combineerror;Error(:,i);];
end
w1 = combineresult(:,1);
w2 = combineresult(:,2);
w3 = combineresult(:,3);
plotpoints(w1,combineerror,'$\vec{w_{1}}$ (m)','Error (mm)','Error - $\vec{w_{1}}$','Error_w1',figureplot);
plotpoints(w2,combineerror,'$\vec{w_{2}}$ (m)','Error (mm)','Error - $\vec{w_{2}}$','Error_w2',figureplot);
plotpointsw3(w3,combineerror,'$\vec{w_{3}}$ (m)','Error (mm)','Error - $\vec{w_{3}}$','Error_w3',figureplot);
meshpoints(w1(15001:16000),w2(15001:16000),combineerror(15001:16000),-0.05,0.05,-
0.05,0.05,'$\vec{w_{1}}$ (m)','$\vec{w_{2}}$ (m)','Error (mm)','Error - $\vec{w_{1}}$ -
$\vec{w_{2}}$','Error_w1_w2',figureplot)

% 2nd Experiment
% Experimenttimes_2 = 51^2;
% Decroator('Generating Random Test Points for 2nd Experiment ...',decroator)
% index = 1;
% for i = 1 : 51
%     for j = 1 : 51
%         Orientation_Given_2(index,1:3) = [-0.05+(i-1)*0.002,-0.05+(j-1)*0.002,0.0499];
%         Orientation_Given_2(index,4:5) = rand(2,1)*2*pi;
%         [Bx,By,Bz] =
generateMFD(Orientation_Given_2(index,:),Sensorposition_w,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Ar
ea_Width,Node_Number);
%         MFD_Given_2(index,:) = Bx;
%         index = index + 1;
%     end
% end
% xlswrite('testpoints_2.xlsx',[Orientation_Given_2,MFD_Given_2]);

Decroator('Reading Test Points for 2nd Experiment ...',decroator);
Test_Points_2 = xlsread('testpoints_2.xlsx');
Orientation_Given_2 = Test_Points_2(:,1:5);
MFD_Given_2 = Test_Points_2(:,6:30)';
Experimenttimes_2 = size(Orientation_Given_2,1);

plottestpoints(Orientation_Given_2(:,1),Orientation_Given_2(:,2),Orientation_Given_2(:,3),'$\vec{w_{1}}$ (m)','$
\vec{w_{2}}$ (m)','$\vec{w_{3}}$ (m)','Test Points 2','Test_Points_2',figureplot);
```

```
% do the 2nd experiments under the best sensordaten and write them in excel-tables
% data = Test_Data(16,:);
% name = ['Experiment_2'];
% experiment(name, data, Experimenttimes_2,
Sensorposition_w,Sensor_number,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number,max_mo
vespeed,max_rotatespeed,Orientation_Given_2,MFD_Given_2);

% analyse the 2nd results
Decroator('Analysing Results 2 ....',decroator);
readdata_2 = xlsread('Experiment_2.xlsx');
exresult_2 = readdata_2(2:end,:);
Error_2 = exresult_2(:,11);

w1_2 = exresult_2(:,1);
w2_2 = exresult_2(:,2);
w3_2 = exresult_2(:,3);
plotpoints(w1_2,Error_2,'$\vec{w_{1}}$ (m)','Error (mm)','Error - $\vec{w_{1}}$','Error_w1_2',figureplot);
plotpoints(w2_2,Error_2,'$\vec{w_{2}}$ (m)','Error (mm)','Error - $\vec{w_{2}}$','Error_w2_2',figureplot);
meshpoints(w1_2,w2_2,Error_2,-0.05,0.05,-0.05,0.05,'$\vec{w_{1}}$ (m)','$\vec{w_{2}}$ (m)','Error (mm)','Error -
$\vec{w_{1}}$ - $\vec{w_{2}}$','Error_w1_w2_2',figureplot)

% 3rd Experiment
% Experimenttimes_3 = 51^2;
% Decroator('Generating Random Test Points for 2nd Experiment ...',decroator)
% index = 1;
% for i = 1 : 51
%     for j = 1 : 51
%         Orientation_Given_3(index,1:3) = [(i-1)*0.0005,(j-1)*0.0005,0.0499];
%         Orientation_Given_3(index,4:5) = rand(2,1)*2*pi;
%         [Bx,By,Bz] =
generateMFD(Orientation_Given_3(index,:),Sensorposition_w,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Ar
ea_Width,Node_Number);
%         MFD_Given_3(index,:) = Bx;
%         index = index + 1;
%     end
% end
% xlswrite('testpoints_3.xlsx',[Orientation_Given_3,MFD_Given_3]);

Decroator('Reading Test Points for 3rd Experiment ...',decroator);
Test_Points_3 = xlsread('testpoints_3.xlsx');
Orientation_Given_3 = Test_Points_3(:,1:5);
MFD_Given_3 = Test_Points_3(:,6:30)';
Experimenttimes_3 = size(Orientation_Given_3,1);

plottestpoints(Orientation_Given_3(:,1),Orientation_Given_3(:,2),Orientation_Given_3(:,3),'$\vec{w_{1}}$ (m)','$
\vec{w_{2}}$ (m)','$\vec{w_{3}}$ (m)','Test Points 3','Test_Points_3',figureplot);

% do the 3rd experiments under the best sensordaten and write them in excel-tables
% data = Test_Data(16,:);
% name = ['Experiment_3'];
% experiment(name, data, Experimenttimes_3,
Sensorposition_w,Sensor_number,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number,max_mo
vespeed,max_rotatespeed,Orientation_Given_3,MFD_Given_3);

% analyse the 3rd results
Decroator('Analysing Results 3 ....',decroator);
readdata_3 = xlsread('Experiment_3.xlsx');
exresult_3 = readdata_3(2:end,:);
Error_3 = exresult_3(:,11);

w1_3 = exresult_3(:,1);
w2_3 = exresult_3(:,2);
w3_3 = exresult_3(:,3);
plotpointsex3(w1_3,Error_3,'$\vec{w_{1}}$ (m)','Error (mm)','Error - $\vec{w_{1}}$','Error_w1_3',figureplot);
plotpointsex3(w2_3,Error_3,'$\vec{w_{2}}$ (m)','Error (mm)','Error - $\vec{w_{2}}$','Error_w2_3',figureplot);
meshpointsex3(w1_3,w2_3,Error_3,0,0.025,0,0.025,'$\vec{w_{1}}$ (m)','$\vec{w_{2}}$ (m)','Error (mm)','Error -
$\vec{w_{1}}$ - $\vec{w_{2}}$','Error_w1_w2_3',figureplot)

% 4th Experiment
Sensor_number_2 = 3;
Sensorposition_w_2 = sensorposition(W(1),Sensor_number_2);
plotsensorposition(Sensorposition_w_2(1,:),Sensorposition_w_2(2,:),'$\vec{w_{1}}$ (m)','$\vec{w_{2}}$ (m)','Sens
or Positions 3','Sensor Positions 3',figureplot);

% Decroator('Generating Random Test Points for 4th Experiment ...',decroator)
% Test_Points_4 = xlsread('testpoints.xlsx');
% Orientation_Given_4 = Test_Points_4(:,1:5);
% for i = 1 : size(Orientation_Given_4,1)
%     [Bx,~,~] =
generateMFD(Orientation_Given_4(i,:),Sensorposition_w_2,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area
```

```
_Width,Node_Number);
%     MFD_Given_4(i,:) = Bx;
% end
% xlswrite('testpoints_4.xlsx',[Orientation_Given_4,MFD_Given_4]);

Decroator('Reading Test Points for 4th Experiment ...',decroator);
Test_Points_4 = xlsread('testpoints_4.xlsx');
Orientation_Given_4 = Test_Points_4(:,1:5);
MFD_Given_4 = Test_Points_4(:,6:14)';
Experimenttimes_4 = size(Orientation_Given_4,1);

% do the 4nd experiments under the best sensordaten and write them in excel-tables
% data = Test_Data(16,:);
% name = ['Experiment_4'];
% experiment(name, data, Experimenttimes_4,
Sensorposition_w_2,Sensor_number_2,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number,ma
x_movespeed,max_rotatespeed,Orientation_Given_4,MFD_Given_4);

% 5th Experiment
Sensor_number_3 = 4;
Sensorposition_w_3 = sensorposition(W(1),Sensor_number_3);
plotsensorposition2(Sensorposition_w_3(1,:),Sensorposition_w_3(2,:),'$\vec{w_{1}}$ (m)','$\vec{w_{2}}$ (m)','Sen
sor Positions 4','Sensor Positions 4',figureplot);

% Decroator('Generating Random Test Points for 5th Experiment ...',decroator)
% Test_Points_5 = xlsread('testpoints.xlsx');
% Orientation_Given_5 = Test_Points_5(:,1:5);
% for i = 1 : size(Orientation_Given_5,1)
%     [Bx,~,~] =
generateMFD(Orientation_Given_5(i,:),Sensorposition_w_3,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area
_Width,Node_Number);
%     MFD_Given_5(i,:) = Bx;
% end
% xlswrite('testpoints_5.xlsx',[Orientation_Given_5,MFD_Given_5]);

Decroator('Reading Test Points for 5th Experiment ...',decroator);
Test_Points_5 = xlsread('testpoints_5.xlsx');
Orientation_Given_5 = Test_Points_5(:,1:5);
MFD_Given_5 = Test_Points_5(:,6:21)';
Experimenttimes_5 = size(Orientation_Given_5,1);

% do the 5th experiments under the best sensordaten and write them in excel-tables
% data = Test_Data(16,:);
% name = ['Experiment_5'];
% experiment(name, data, Experimenttimes_5,
Sensorposition_w_3,Sensor_number_3,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number,ma
x_movespeed,max_rotatespeed,Orientation_Given_5,MFD_Given_5);

% 6th Experiment
Sensor_number_4 = 7;
Sensorposition_w_4 = sensorposition(W(1),Sensor_number_4);
plotsensorposition3(Sensorposition_w_4(1,:),Sensorposition_w_4(2,:),'$\vec{w_{1}}$ (m)','$\vec{w_{2}}$ (m)','Sen
sor Positions 7','Sensor Positions 7',figureplot);

% Decroator('Generating Random Test Points for 5th Experiment ...',decroator)
% Test_Points_6 = xlsread('testpoints.xlsx');
% Orientation_Given_6 = Test_Points_6(:,1:5);
% for i = 1 : size(Orientation_Given_6,1)
%     [Bx,~,~] =
generateMFD(Orientation_Given_6(i,:),Sensorposition_w_4,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area
_Width,Node_Number);
%     MFD_Given_6(i,:) = Bx;
% end
% xlswrite('testpoints_6.xlsx',[Orientation_Given_6,MFD_Given_6]);

Decroator('Reading Test Points for 6th Experiment ...',decroator);
Test_Points_6 = xlsread('testpoints_6.xlsx');
Orientation_Given_6 = Test_Points_6(:,1:5);
MFD_Given_6 = Test_Points_6(:,6:54)';
Experimenttimes_6 = size(Orientation_Given_6,1);

% do the 6th experiments under the best sensordaten and write them in excel-tables
% data = Test_Data(16,:);
% name = ['Experiment_6'];
% experiment(name, data, Experimenttimes_6,
Sensorposition_w_4,Sensor_number_4,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number,ma
x_movespeed,max_rotatespeed,Orientation_Given_6,MFD_Given_6);

% analyse 4-6th Experiments
meanerror_array = zeros(1,4);
```

```
for i = 1 : 3
    Experiment_numer = ['4','5','6'];
    name = ['Experiment_',Experiment_numer(i),'.xlsx'];
    data = xlsread(name);
    ERROR = data(2:end,11);
    for k = 1 : 1000
        if ERROR(k) > 5
            ERROR(k) = NaN;
        end
    end
    meanerror_array(i) = nanmean(ERROR);
end
data = xlsread('Ex_15.xlsx');
ERROR = data(2:end,11);
for k = 1 : 1000
    if ERROR(k) > 5
        ERROR(k) = NaN;
    end
end
meanerror_array(4) = nanmean(ERROR);
temp = meanerror_array(3);
meanerror_array(3) = meanerror_array(4);
meanerror_array(4) = temp;
plotarrayresult(meanerror_array,figureplot)


% Hardware Validation
% x_v = [0 -0.03 0 0.02];
% y_v = [0 -0.04 0.01 0.05];
% z_v = [0.05 0.02 0.03 0.04];
% theta1_v = [0 0 pi/2 -pi/2];
% theta2_v = [0 pi/4 pi/6 -pi/6];
% for i = 1 : 4
%     Orientation_Validation(i,:) = [x_v(i) y_v(i) z_v(i) theta1_v(i) theta2_v(i)];
%     [Bx,~,~] =
generateMFD(Orientation_Validation(i,:),Sensorposition_w_3,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,A
rea_Width,Node_Number);
%     MFD_Validation(i,:) = Bx;
% end
% Orientation_Validation(:,4:5) = Orientation_Validation(:,4:5) / pi * 180;
% xlswrite('Validation.xlsx',[Orientation_Validation,MFD_Validation]);

Validation_Data = xlsread('Validation.xlsx');
Orientation_Validation = Validation_Data(:,1:5);
Orientation_Validation_show = Orientation_Validation;
Orientation_Validation(:,4:5) = Orientation_Validation(:,4:5) * pi / 180;
for i = 1 : 4
    name = ['Orientation ',num2str(i)];
    orientation_indicator(Orientation_Validation(i,:),name,['Validation_',num2str(i)],2)
end
MFD_Validation = Validation_Data(:,6:21);

validationresult = xlsread('Validationresult.xlsx');
zero_voltage = validationresult(:,2);
validation_evaluation = zeros(k,1);
voltage_theory_quantization = zeros(16,4);
for k = 1 : 4
    voltage_is = validationresult(:,2*k+1);
    MFD_theory = validationresult(:,2*k+2);
    voltage_difference_theory = MFD_theory * 5 / 3000;
    voltage_theory = voltage_difference_theory + zero_voltage;
    voltage_theory_quantization(:,k) = (floor((voltage_theory + 5/2048) / (5/1024))) * 5 / 1024;
    validation_evaluation(k) = norm(voltage_is - voltage_theory_quantization(:,k));
end
validation_evaluation =
[NaN,validation_evaluation(1),NaN,validation_evaluation(2),NaN,validation_evaluation(3),NaN,validation_evaluatio
n(4)]/16;
% xlswrite('validation_evaluation.xlsx', validation_evaluation)

xlswrite('validation_evaluation.xlsx',[validationresult(:,3),voltage_theory_quantization(:,1),validationresult(:
,5),voltage_theory_quantization(:,2),validationresult(:,7),voltage_theory_quantization(:,3),validationresult(:,9
),voltage_theory_quantization(:,4);validation_evaluation])


% important!!
% if the figures are already with tranparent background, the parameter
% trans must be set to 0 !!!!!!!!!!!
% Decroator('Processing Pictures ....',decroator);
% trans = 0;
% files = dir('E:\Matlab\Project_github\MTS');
```

```
% for i = 1 : size(files,1)
%     NAME = files(i).name;
%     l = size(NAME,2);
%     if l >= 4
%         TYPE = NAME(l-2:l);
%         if TYPE == 'png'
%             [I alpha] = cutfigure(NAME);
%             if trans == 1
%                 imwrite(I,NAME,'Alpha',alpha)
%             elseif trans == 0
%                 imwrite(I,NAME)
%             end
%         end
%     end
% end

Decroator('Programm finished.',decroator);
```

8.2 Magnetfield.m

```
function [MFS_y,MFS_z,Coordinate_q2,Coordinate_q3] =
Magnetfield(Magnet_radius,Magnet_altitude,Magnet_remanenz,Area_Width,Area_Length,Node_Number,filename,write)
% m is the magnetization of the Magnets. Dor Permanetmagnet is m = Br/mu_0
m = Magnet_remanenz / (4*pi*10^(-7));

h12 = zeros(Node_Number + 1,Node_Number * 2 + 1);
h13 = h12;
h32 = h12;
h33 = h12;

% Megnetic Field from area 1
for i = 1 : Node_Number + 1
    t1 = clock;
    for j = 1 : Node_Number * 2 + 1
        Coordinate_q2 = (i - 1) * Area_Width / Node_Number;
        Coordinate_q3 = (j - 1) * Area_Length / 2 / Node_Number - Area_Length/2;
        h12(i,j) = H12(Coordinate_q2,Coordinate_q3,m,Magnet_radius,Magnet_altitude);
        h13(i,j) = H13(Coordinate_q2,Coordinate_q3,m,Magnet_radius,Magnet_altitude);
    end
    t2 = clock;
    delta_t = etime(t2,t1);
    rest_time = delta_t * (Node_Number+1-i);
    percentage = floor(i/(Node_Number+1)*10000)/100;
    Decroator(['Calculating MFS ',num2str(percentage),'%. Estimated to be finished in
',num2str(floor(rest_time)),' seconds ...'],1);
end

% Megnetic Field from area 3
for i = 1 : Node_Number + 1
    for j = 1 : Node_Number * 2 + 1
        h32(i,j) = - h12(i,2*Node_Number+2-j);
        h33(i,j) = h13(i,2*Node_Number+2-j);
    end
end

MFS_y = h12 + h32;
MFS_z = h13 + h33;

% let the magnetic field strength inside the Manget to 0
for i = 1 : Node_Number + 1
    for j = 1 : Node_Number * 2 + 1
        Coordinate_q2 = (i - 1) * Area_Width / Node_Number;
        Coordinate_q3 = (j - 1) * Area_Length / 2 / Node_Number - Area_Length/2;
        if and((Coordinate_q2 <= Magnet_radius + Area_Width/Node_Number),(Coordinate_q3 <=
Magnet_altitude/2+Area_Length/2/Node_Number))
            MFS_y(i,j) = NaN;
            MFS_z(i,j) = NaN;
        end
    end
end

%Symmetry to eliminate the numerical error on coordinate
for i = 1 : Node_Number + 1
    for j = 1 : Node_Number
        MFS_y(i,j) = - MFS_y(i,2*Node_Number+2-j);
        MFS_z(i,j) = MFS_z(i,2*Node_Number+2-j);
    end
end

i = 1 : Node_Number + 1;
j = 1 : Node_Number * 2 + 1;
Coordinate_q2 = (i - 1) * Area_Width / Node_Number;
Coordinate_q3 = (j - 1) * Area_Length / 2 / Node_Number - Area_Length/2;

if write == 1
    xlswrite(['Hy_',filename,'.xlsx'],MFS_y);
    xlswrite(['Hz_',filename,'.xlsx'],MFS_z);
    xlswrite(['q2_',filename,'.xlsx'],Coordinate_q2);
    xlswrite(['q3_',filename,'.xlsx'],Coordinate_q3);
end
```

8.3 H12.m

```
function S = H12(varargin)
if length(varargin) == 5
    n = 100;
else
    n = varargin{6};
end
y = varargin{1};
z = varargin{2};
magnetization = varargin{3};
Magnet_radius = varargin{4};
Magnet_altitude = varargin{5};

dr = Magnet_radius/n;
dt = 2*pi/n;

sum = 0;
for r = 0 : dr : Magnet_radius
    for theta = 0 : dt :2*pi
        sum = sum + (y-r*sin(theta))/((r*cos(theta))^2+(y-r*sin(theta))^2+(z-Magnet_altitude/2)^2)^(3/2)*r*dr*dt;
    end
end
S = sum*magnetization/4/pi;
```

8.4 H13.m

```
function S = H13(varargin)
if length(varargin) == 5
    n = 100;
else
    n = varargin{6};
end
y = varargin{1};
z = varargin{2};
magnetization = varargin{3};
Magnet_radius = varargin{4};
Magnet_altitude = varargin{5};


dr = Magnet_radius/n;
dt = 2*pi/100;

sum = 0;
for r = 0 : dr : Magnet_radius
    for theta = 0 : dt :2*pi
        sum = sum + (z-Magnet_altitude/2)/((r*cos(theta))^2+(y-r*sin(theta))^2+(z-
Magnet_altitude/2)^2)^(3/2)*r*dr*dt;
    end
end
S = sum*magnetization/4/pi;
```

8.5 grad.m

```
function gradient = grad(MFD,Coordinate_q2,Coordinate_q3)
% to calculate the gradient
dq2 = Coordinate_q2(2) - Coordinate_q2(1);
dq3 = Coordinate_q3(2) - Coordinate_q3(1);
for i = 1 : length(Coordinate_q2) - 1
    for j = 1 : length(Coordinate_q3) - 1
        gradient(i,j,1) = (MFD(i+1,j) - MFD(i,j))/dq2;
        gradient(i,j,2) = (MFD(i,j+1) - MFD(i,j))/dq3;
    end
end
```

8.6 sensorposition.m

```
function Sensor = sensorposition(Workspace_size,Sensor_number)
N = Sensor_number * Sensor_number;
Sensor = zeros(3,N);

for i = 0 : N-1
    a = mod(i,Sensor_number);
    b = (i - a) / Sensor_number;
    Sensor(2,i+1) = a * Workspace_size/(Sensor_number-1) - Workspace_size/2;
    Sensor(1,i+1) = - b * Workspace_size/(Sensor_number-1) + Workspace_size/2;
end
```

8.7 coordinatew2i.m

```
function [rcs, thetak] = coordinatew2i(Orientation,Sensor_position)
% rcs[1,:] is the Coordinates of the Sensors in Magentcoordinaten
% Orientation is the position and rotation of the magnet, c1,c2,c3,theta1,theta2
% Sensor is the Position of Sensors
% rcs[2,:] is the Coordinaten of Vergleichspunkt

c1 = Orientation(1);
c2 = Orientation(2);
c3 = Orientation(3);
theta1 = Orientation(4);
theta2 = Orientation(5);

s1 = Sensor_position(1);
s2 = Sensor_position(2);

rcs(1,1) = (-c1+s1)*cos(theta2) + (-c2+s2)*sin(theta1)*sin(theta2) + c3*cos(theta1)*sin(theta2);
rcs(1,2) = (-c2+s2)*cos(theta1) - c3*sin(theta1);
rcs(1,3) = (-c1+s1)*sin(theta2) - (-c2+s2)*sin(theta1)*cos(theta2) - c3*cos(theta1)*cos(theta2);

if rcs(1,2) > 0
    thetak = - atan(rcs(1,1)/rcs(1,2));
elseif rcs(1,2) < 0
    if rcs(1,1) <= 0
        thetak = - atan(rcs(1,1)/rcs(1,2)) + pi;
    else
        thetak = - atan(rcs(1,1)/rcs(1,2)) - pi;
    end
else
    if rcs(1,1) > 0
        thetak = - pi/2;
    else
        thetak =  pi/2;
    end
end
if rcs(1,1) == 0 && rcs(1,2) == 0
    thetak = 0;
    rcs(2,2) = 0;
end
rcs(2,1) = 0;
rcs(2,2) = -rcs(1,1)*sin(thetak) + rcs(1,2)*cos(thetak);
rcs(2,3) = rcs(1,3);
```

8.8 itplt.m

```
function [HyV,HzV] = itplt(x,y,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number)
Vy1 = floor(x/Area_Width*Node_Number+1);
Vz1 = floor((y+Area_Length/2)/Area_Length*2*Node_Number+1);
Vy2 = Vy1 + 1;
Vz2 = Vz1 + 1;

x1 = Coordinate_q2(Vy1);
x2 = Coordinate_q2(Vy2);
y1 = Coordinate_q3(Vz1);
y2 = Coordinate_q3(Vz2);
f11 = MFD_y(Vy1,Vz1);
f12 = MFD_y(Vy1,Vz2);
f21 = MFD_y(Vy2,Vz1);
f22 = MFD_y(Vy2,Vz2);
g11 = MFD_z(Vy1,Vz1);
g12 = MFD_z(Vy1,Vz2);
g21 = MFD_z(Vy2,Vz1);
g22 = MFD_z(Vy2,Vz2);

HyV = 1/(x2-x1)/(y2-y1) * [x2-x,x-x1] * [f11,f12;f21,f22] * [y2-y,y-y1]';
HzV = 1/(x2-x1)/(y2-y1) * [x2-x,x-x1] * [g11,g12;g21,g22] * [y2-y,y-y1]';
```

8.9 coordinatei2w.m

```
function [HxS, HyS, HzS] = coordinatei2w(theta1,theta2,thetak,HyV,HzV)
HxS = -HyV*sin(thetak)*cos(theta2) + HzV*sin(theta2);
HyS = -HyV*sin(thetak)*sin(theta1)*sin(theta2)+HyV*cos(thetak)*cos(theta1) - HzV*sin(theta1)*cos(theta2);
HzS = HyV*sin(thetak)*cos(theta1)*sin(theta2)+HyV*cos(thetak)*sin(theta1) + HzV*cos(theta1)*cos(theta2);
```

8.10 out_of_range.m

```
function D1 = out_of_range(Coordinate_q2,Coordinate_q3,Sensor_data,MFD_value)
D1 = zeros(length(Coordinate_q2),length(Coordinate_q3));
for i = 1 : length(Coordinate_q2)
    for j = 1 : length(Coordinate_q3)
        if MFD_value(i,j) > Sensor_data(3)
            D1(i,j) = 1;
        elseif isnan(MFD_value(i,j)) == 1
            D1(i,j) = 1;
        else
            D1(i,j) = 0;
        end
    end
end
```

8.11 detectablearea.m

```
function D = detectablearea(Coordinate_q2,Coordinate_q3,D1,k,Sensitivity, GBm)
D2 = zeros(length(Coordinate_q2),length(Coordinate_q3));
% detectable position change in m
a = 1/1000;
for i = 1: size(GBm,1)
    for j = 1: size(GBm,2)
        if sqrt(GBm(i,j,1)^2+GBm(i,j,2)^2) < 5 / a / 2^k / Sensitivity
            D2(i,j) = 1;
        else
            D2(i,j) = NaN;
        end
    end
end
D = zeros(length(Coordinate_q2),length(Coordinate_q3));
for i = 1 : length(Coordinate_q2)
    for j = 1 : length(Coordinate_q3)
        if D1(i,j)==1 ||  D2(i,j)==1
            D(i,j) = 1;
        else
            D(i,j) = 0;
        end
    end
end
```

8.12 detectmovement.m

```
function a = detectmovement(Coordinate_q2,Coordinate_q3,Sensitivity,k,GBm)
a = NaN(length(Coordinate_q2),length(Coordinate_q3));
for i = 1 : size(GBm,1)
    for j = 1 : size(GBm,2)
        a(i,j) = 5/2^k/Sensitivity/sqrt(GBm(i,j,1)^2+GBm(i,j,2)^2);
    end
end
a = a*1000;
```

8.13 randpoint.m

```
function [O Bb] = randpoint(Sensorposition_w, Sensornumber, Coordinate_q2, Coordinate_q3, MFD_y, MFD_z,
Area_Length, Area_Width, Node_Number)
flag = 0;
while flag == 0
    O(1) = rand()*0.1-0.05;
    O(2) = rand()*0.1-0.05;
    O(3) = rand()*0.05;
    O(4) = rand()*2*pi;
    O(5) = rand()*2*pi;
    [Bx,By,Bz] =
generateMFD(O,Sensorposition_w,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number);
    Bb = Bx;
    collision = isnan(sum(Bb));
    if collision == 0
        flag = 1;
    end
end
```

8.14 generateMFD.m

```
function [Bx,By,Bz] =
generateMFD(Orientation_Given,Sensorposition_w,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,No
de_Number)
for s = 1 : length(Sensorposition_w)
```

```
    [r_cs, theta_k]= coordinatew2i(Orientation_Given,Sensorposition_w(:,s));
    [MFD_y_Vglpunkt,MFD_z_Vglpunkt] =
itplt(r_cs(2,2),r_cs(2,3),Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number);
    [MFD_at_Sensor_x, MFD_at_Sensor_y, MFD_at_Sensor_z] =
coordinatei2w(Orientation_Given(3),Orientation_Given(4),theta_k,MFD_y_Vglpunkt,MFD_z_Vglpunkt);
    Bx(s) =  MFD_at_Sensor_x;
    By(s) =  MFD_at_Sensor_y;
    Bz(s) =  MFD_at_Sensor_z;
end
```

8.15 experiment.m

```
function experiment(name, Test_Data, experimenttimes, Sensorposition_w, Sensor_number, Coordinate_q2,
Coordinate_q3, MFD_y, MFD_z, Area_Length, Area_Width, Node_Number, vmax, rmax, OG, BG)
Sensordata = Test_Data(1:4);
k = Test_Data(5);
Os = zeros(experimenttimes,5);
Error = zeros(experimenttimes,2);
for n = 1 : experimenttimes
    if n == 1
        Decroator(['Localizing the 1st Point / ',num2str(experimenttimes),' Points in ',name],1)
    elseif n == 2
        Decroator(['Localizing the 2nd Point / ',num2str(experimenttimes),' Points in ',name],1)
    elseif n == 3
        Decroator(['Localizing the 3rd Point / ',num2str(experimenttimes),' Points in ',name],1)
    else
        Decroator(['Localizing the ',num2str(n),'th Point / ',num2str(experimenttimes),' Points in ',name],1)
    end
    Og = OG(n,:);
    Bg = BG(:,n);
    Vn = Noising(Bg,Sensordata);
    Bb = quantize(Vn,k,Sensordata);
    initialspace(1) = max(Og(1)-vmax*0.01,-0.05);
    initialspace(2) = min(Og(1)+vmax*0.01,0.05);
    initialspace(3) = max(Og(2)-vmax*0.01,-0.05);
    initialspace(4) = min(Og(2)+vmax*0.01,0.05);
    initialspace(5) = max(Og(3)-vmax*0.01,0);
    initialspace(6) = min(Og(3)+vmax*0.01,0.05);
    initialspace(7) = max(Og(4)-rmax*0.01,0);
    initialspace(8) = min(Og(4)+rmax*0.01,2*pi);
    initialspace(9) = max(Og(5)-rmax*0.01,0);
    initialspace(10) = min(Og(5)+rmax*0.01,2*pi);
    node = 4;
    Os(n,:) =
localization2(Bb,initialspace,node,Sensor_number,Sensorposition_w,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_L
ength,Area_Width,Node_Number);
    Error(n,1) = norm(Os(n,1:3) - OG(n,1:3))*1000;
    Error(n,1)
    Error(n,2) = norm(Os(n,4:5) - OG(n,4:5));
end
Parameter = [Test_Data,NaN(1,7)];
Result = [Parameter;OG(1:experimenttimes,:),Os,Error];
name = [name,'.xlsx'];
xlswrite(name,Result);
```

8.16 Noising.m

```
function [f,o] = Noising(MFD,Sensordata)
% o = 0: the MFD is out of the range.
% MFD is the magnetic flux density
% error is the error in sensitivity in percent for eg 1%: error = 1
% ZV is the zero voltage
% Range is the range of the sensor
% Sen is the sensitivity in V/G
error = Sensordata(1);
ZV = Sensordata(2);
Range = Sensordata(3);
Sen = Sensordata(4);
f = zeros(length(MFD),1);
for i = 1: length(MFD)
    if abs(MFD(i)) > Range
        o(i) = 0;
    else
        o(i) = 1;
    end
    f(i) = ((rand()-0.5)*2*error/100+1) * Sen * MFD(i) + ZV;
end
```

8.17 quantize.m

```
function Bb = quantize(Vn, k, Sensordata)
Vn = Vn + 1/2^(k+1);
Vq = 5/2^k*floor(Vn*2^k/5);
Bb = (Vq-Sensordata(2))/Sensordata(4);
```

8.18 localization2.m

```
function Os = localization2(Bb,initialspace, node,
Sensor_number,Sensorposition_w,Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number)
spacelength = initialspace(2)-initialspace(1);
BS = zeros(Sensor_number^2,node,node,node,node,node);
c1 = zeros(node); c2 = zeros(node); c3 = zeros(node); theta1 = zeros(node); theta2 = zeros(node);
C = zeros(1,5); G = zeros(node,node,node,node,node);
while spacelength > 0.0003
     for s = 1:Sensor_number^2
        for i = 1:node
            for j = 1:node
                for k = 1:node
                    for l = 1:node
                        for m = 1:node
                            c1(i) = initialspace(1) + (i-1)*(initialspace(2)-initialspace(1))/(node-1);
                            c2(j) = initialspace(3) + (j-1)*(initialspace(4)-initialspace(3))/(node-1);
                            c3(k) = initialspace(5) + (k-1)*(initialspace(6)-initialspace(5))/(node-1);
                            theta1(l) = initialspace(7) + (l-1)*(initialspace(8)-initialspace(7))/(node-1);
                            theta2(m) = initialspace(9) + (m-1)*(initialspace(10)-initialspace(9))/(node-1);
                            C = [c1(i),c2(j),c3(k),theta1(l),theta2(m)];
                            [rcs, thetak]= coordinatew2i(C,Sensorposition_w(:,s));
                            [ByV,BzV] =
itplt(rcs(2,2),rcs(2,3),Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number);
                            [BxS, ~, ~] = coordinatei2w(C(3),C(4),thetak,ByV,BzV);
                            BS(s,i,j,k,l,m) = BxS;
                        end
                    end
                end
            end
        end
    end
    for i = 1:node
        for j = 1:node
            for k = 1:node
                for l = 1:node
                    for m = 1:node
                       G(i,j,k,l,m) = nansum((BS(:,i,j,k,l,m) - Bb).^2);
                    end
                end
            end
        end
    end
    temp = G(1,1,1,1,1);    INDEX = [1,1,1,1,1];
    for i = 1:node
        for j = 1:node
            for k = 1:node
                for l = 1:node
                    for m = 1:node
                        if G(i,j,k,l,m) < temp
                            temp = G(i,j,k,l,m); INDEX = [i,j,k,l,m];
                        end
                    end
                end
            end
        end
    end
    Os = [c1(INDEX(1)),c2(INDEX(2)),c3(INDEX(3)),theta1(INDEX(4)),theta2(INDEX(5))];
    spacelength = (initialspace(2)-initialspace(1))/(node-1);
    rotatespacelength = (initialspace(8)-initialspace(7))/(node-1);
    initialspace(1) = max(Os(1)-spacelength,-0.05);
    initialspace(2) = min(Os(1)+spacelength,0.05);
    initialspace(3) = max(Os(2)-spacelength,-0.05);
    initialspace(4) = min(Os(2)+spacelength,0.05);
    initialspace(5) = max(Os(3)-spacelength,0);
    initialspace(6) = min(Os(3)+spacelength,0.05);
    initialspace(7) = max(Os(4)-rotatespacelength,0);
    initialspace(8) = min(Os(4)+rotatespacelength,2*pi);
    initialspace(9) = max(Os(5)-rotatespacelength,0);
    initialspace(10) = min(Os(5)+rotatespacelength,2*pi);
end
```

8.19 localization.m

```
function Os = localization(MFD_detected, initial_Orientation, MFD_y, MFD_z, GBy, GBz, Coordinate_q2,
Coordinate_q3, Area_Length, Area_Width, Node_Number, Sensor_number, Sensorposition_w)
for Node_Number = 1 : 30
    for j = 1 : Sensor_number^2
        [rcs, thetak]= coordinatew2i(initial_Orientation,Sensorposition_w(:,j));
        [ByV,BzV] =
itplt(rcs(2,2),rcs(2,3),Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number);
        [BxS, ByS, BzS] = coordinatei2w(initial_Orientation(3),initial_Orientation(4),thetak,ByV,BzV);
        BxO0 = BxS;
        [GB22V,GB32V] =
itplt(rcs(2,2),rcs(2,3),Coordinate_q2,Coordinate_q3,GBy(:,:,1),GBz(:,:,1),Area_Length,Area_Width,Node_Num
ber);
        [GB23V,GB33V] =
itplt(rcs(2,2),rcs(2,3),Coordinate_q2,Coordinate_q3,GBy(:,:,2),GBz(:,:,2),Area_Length,Area_Width,Node_Num
ber);
        A = [-cos(initial_Orientation(5)),0,-sin(initial_Orientation(4));
            -sin(initial_Orientation(4))*sin(initial_Orientation(5)),-
cos(initial_Orientation(4)),sin(initial_Orientation(4))*cos(initial_Orientation(5));
            cos(initial_Orientation(4))*sin(initial_Orientation(5)),-sin(initial_Orientation(4)),-
cos(initial_Orientation(4))*cos(initial_Orientation(5));
            (-
initial_Orientation(2)+Sensorposition_w(2,j))*cos(initial_Orientation(4))*sin(initial_Orientation(5))-
initial_Orientation(3)*sin(initial_Orientation(4))*sin(initial_Orientation(5)),...
                (initial_Orientation(2)-Sensorposition_w(2,j))*sin(initial_Orientation(4))-
initial_Orientation(3)*cos(initial_Orientation(4)),...
                (initial_Orientation(2)-
Sensorposition_w(2,j))*cos(initial_Orientation(4))*cos(initial_Orientation(5))+initial_Orientation(3)*sin
(initial_Orientation(4))*cos(initial_Orientation(5));
            (initial_Orientation(1)-Sensorposition_w(1,j))*sin(initial_Orientation(5))+(-
initial_Orientation(2)+Sensorposition_w(2,j))*sin(initial_Orientation(4))*cos(initial_Orientation(5))+ini
tial_Orientation(3)*cos(initial_Orientation(4))*cos(initial_Orientation(5)),...
                0,...
                (-initial_Orientation(1)+Sensorposition_w(1,j))*cos(initial_Orientation(5))+(-
initial_Orientation(2)+Sensorposition_w(2,j))*sin(initial_Orientation(4))*sin(initial_Orientation(5))+ini
tial_Orientation(3)*cos(initial_Orientation(4))*sin(initial_Orientation(5))];
       pj(:,j) = (MFD_detected(j)-BxO0) * A * [0,-sin(thetak),0;0,cos(thetak),0;0,0,1] *
[0,0,0;0,GB22V,GB32V;0,GB23V,GB33V] * [0;-
sin(thetak)*cos(initial_Orientation(5));sin(initial_Orientation(5))];
       p = nansum(pj')';
       p = p / norm(p);
    end
    alpha(1) = 0.0002;
    for i = 2 : 20
        alpha(i) = alpha(i-1) * 1.7;
    end
    for i = 1 : 10
        O1 = initial_Orientation + p * alpha(i);
        O1(1) = min(O1(1),0.05);
        O1(1) = max(O1(1),-0.05);
        O1(2) = min(O1(2),0.05);
        O1(2) = max(O1(2),-0.05);
        O1(3) = min(O1(3),0.05);
        O1(3) = max(O1(3),0);
        for j = 1 : Sensor_number^2
            [rcs, thetak]= coordinatew2i(O1,Sensorposition_w(:,j));
            [ByV,BzV] =
itplt(rcs(2,2),rcs(2,3),Coordinate_q2,Coordinate_q3,MFD_y,MFD_z,Area_Length,Area_Width,Node_Number);
            [BxS, ByS, BzS] = coordinatei2w(O1(3),O1(4),thetak,ByV,BzV);
            BxO0 = BxS;
            Gj(j) = 0.5 * norm(MFD_detected(j)-BxS)^2;
        end
        G(i) = nansum(Gj);
        if i >= 3
            if G(i-2)> G(i-1) & G(i-1) < G(i)
                initial_Orientation = initial_Orientation + alpha(i-1)*p;
                break
            end
        end
    end
end
Os = initial_Orientation;
```

8.20 Decorator.m

```
function Decroator(Executing,decroator)
if decroator == 1
    Executing
end
```

8.21 orientation_indicator.m

```
function orientation_indicator(Orientation,name, filename, figureplot)
if figureplot >= 1
    vec_i_1 = [0,0,0.02];
    n = 100;
    r = 0.009;
    for i = 1 : n
        theta = 2*pi/n * i;
        circle_i(i,:) = [r * sin(theta), r * cos(theta),0] ;
    end

    theta1 = Orientation(4);
    theta2 = Orientation(5);
    mwi = [cos(theta2) 0 sin(theta2);...
        sin(theta1)*sin(theta2) cos(theta1) -sin(theta1)*cos(theta2);...
        -cos(theta1)*sin(theta2) sin(theta1) cos(theta1)*cos(theta2)];

    vec_w_1 = mwi * vec_i_1';
    for i = 1 : n
        circle_w(i,:) = (mwi * circle_i(i,:)')' ;
    end

    O_w = Orientation(1:3)';
    E_w_1 = O_w + vec_w_1;
    figure
    set(gcf,'outerposition',get(0,'screensize'));
    plot3([O_w(1),E_w_1(1)],[O_w(2),E_w_1(2)],[O_w(3),E_w_1(3)],'r-','LineWidth',5)
    for i = 1 : n
        hold on
        plot3(circle_w(i,1)+O_w(1),circle_w(i,2)+O_w(2),circle_w(i,3)+O_w(3),'k.')
    end
    hold on
    plot3(O_w(1),O_w(2),O_w(3),'k.','Markersize',30)
    hold on
    plot3([O_w(1),-0.075],[O_w(2),O_w(2)],[O_w(3),O_w(3)],'b-','LineWidth',2)
    hold on
    plot3([O_w(1),O_w(1)],[O_w(2),-0.075],[O_w(3),O_w(3)],'b-','LineWidth',2)
    hold on
    plot3([O_w(1),O_w(1)],[O_w(2),O_w(2)],[O_w(3),0],'b-','LineWidth',2)

    set(gca,'Fontsize',20)
    set(gca,'fontname','times new Roman')
    T = title(name,'fontsize',40);
    set(T,'Interpreter','latex')
    T = xlabel('$\vec{w_{1}}$ (m)','fontsize',30);
    set(T,'Interpreter','latex')
    T = ylabel('$\vec{w_{2}}$ (m)','fontsize',30);
    set(T,'Interpreter','latex')
    T = zlabel('$\vec{w_{3}}$ (m)','fontsize',30);
    set(T,'Interpreter','latex')

    xlim([-0.075 0.075])
    ylim([-0.075 0.075])
    zlim([0 0.075])
    pbaspect([1 1 0.5])
    xticks([-0.075,-0.05,-0.025,0,0.025,0.05,0.075])
    yticks([-0.075,-0.05,-0.025,0,0.025,0.05,0.075])
    zticks([0,0.025,0.05 0.075])
    xticklabels({'-0.075','-0.05','-0.025','0','0.025','0.05','0.075'})
    yticklabels({'-0.075','-0.05','-0.025','0','0.025','0.05','0.075'})
    zticklabels({'0','0.025','0.05','0.075'})
    grid on
    drawnow
    if figureplot == 2
        print([filename,'.png'],'-dpng','-r600')
    end
end
```

8.22 cuttingfigure.m

```
function [Inew alpha] = cutfigure(filename,trans)
Decroator(['Processing "',filename,'" ....'],1);
I = imread(filename);
Ig = rgb2gray(I);
[m,n] = size(Ig);
cutedge = [1 1 m n];
for i = 1 : m
    if sum(Ig(i,:)) < n * 255
        cutedge(1) = i;
        break
    end
end
for i = 1 : n
    if sum(Ig(:,i)) < m * 255
        cutedge(2) = i;
        break
    end
end
for i = m : -1 : 1
    if sum(Ig(i,:)) < n * 255
        cutedge(3) = i;
        break
    end
end
for i = n : -1 : 1
    if sum(Ig(:,i)) < m * 255
        cutedge(4) = i;
        break
    end
end
Cutedge = cutedge + [-100 -100 100 100];
Cutedge(1) = max(Cutedge(1),1)
Cutedge(2) = max(Cutedge(2),1)
Cutedge(3) = min(Cutedge(3),m)
Cutedge(4) = min(Cutedge(4),n)

Inew = I(Cutedge(1):Cutedge(3),Cutedge(2):Cutedge(4),:);

[M,N] = size(Ig);
alpha = zeros(M,N);
if trans == 1
    for k = 1 : M
        for t = 1 : N
            if Inew(k,t) == 255
                alpha(k,t) = 0;
            else
                alpha(k,t) = 1;
            end
        end
    end
end
```

8.23 Arduino Code

```
void setup() {
  pinMode(13, OUTPUT);
  digitalWrite(13, 1);
  Serial.begin(9600);
}
void loop() {
  char pins[16] = {A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15};
  int sensor[16];
  double voltages[16];
  for (int i = 0; i <=15; i++)
    {
    sensor[i] = analogRead(pins[i]);
    voltages[i] = sensor[i] * (5.0 / 1023.0);
    Serial.println(i+1);
    Serial.println(voltages[i],4);
    }
}
```