Projekt 17

(Fehlerdetektion und Gesichtserkennung; roor, kirs)

Dieses Projekt führt Sie zur sogenannten Singulärwertzerlegung von Matrizen (SVD für "singular value decomposition"). Diese Methode der linearen Algebra hat unzählige Anwendungen in der Technik und insbesondere im *machine learning.* Zudem ist sie verwandt mit der Eigenwertzerlegung, die Sie aus anderen Mathematikmodulen kennen.

Sie werden im Laufe der Projektbearbeitung in überschaubaren Schritten an die SVD herangeführt. Dabei lernen Sie, diese Methode *geometrisch* zu interpretieren, wo sich ein Berührungspunkt mit der Ausgleichsrechnung ergibt. Schliesslich erarbeiten Sie *eine* von zwei konkreten Anwendungen, die *Fehlerdetektion* in einem Sensorsystem oder die *Gesichtserkennung* bei der Bildbearbeitung.

Anwendungen der Singulärwertzerlegung Die Singulärwertzerlegung einer beliebigen Matrix kann mit der Formel $\underline{\boldsymbol{M}} = \underline{\boldsymbol{U}}\underline{\boldsymbol{\Sigma}}\underline{\boldsymbol{V}}^\top$ beschrieben werden. Dabei sind $\underline{\boldsymbol{U}}$ und $\underline{\boldsymbol{V}}$ orthogonale Matrizen und $\underline{\boldsymbol{\Sigma}}$ ist eine Diagonalmatrix. Es bestehen Analogien zur QR-Zerlegung und zur Eigenwertzerlegung, jedoch ist die SVD viel breiter anwendbar. Zuerst werden typische Anwendungsfälle beschrieben, die Sie in der Folge selbständig bearbeiten.

- Lösung folgender Aufgabe: Finde eine Gerade zu gegebenen Punkten, so dass die Summe der Quadrate der *euklidschen Abstände* von den Punkten zur Gerade minimal wird. Der entscheidende Punkt: minimiert wird der *rechtwinklige* Abstand zur Gerade, nicht die vertikale Abweichung wie bei der Ausgleichsrechnung ($\rightsquigarrow$ orthogonale Regression)!

- Aufbau eines Fehlerdetektors für einen Winkelsensor: Dabei wird Information in redundanten Messungen des Winkels via Magnetfeld genutzt, um eine unerwünschte Änderung der Kalibration *ohne* Referenzmessungen zu detektieren. Diese Aufgabe stellt sich z.B. bei der Messung der Bremspedalstellung, wo solche Sensorsysteme absolut sicherheitskritisch sind.

- Detektion von Gesichtern in Bildern: Ihr Smartphone ist wahrscheinlich mit dieser Funktion ausgerüstet, um das Scharfstellen beim Selfie zu optimieren. Wie das mit geringem Aufwand möglich ist, zeigt die letzte Aufgabe dieses Projekts.

Singulärwertzerlegung Eine Einführung in die Singurlärwertzerlegung finden Sie z.B. in SVD (auf Deutsch) oder – wesentlich informativer – in SVD (auf Englisch). Die Ihnen bekannte Eigenwertzerlegung einer Matrix $\underline{\boldsymbol{A}}$ mit reellen Eigenwerten ist nur für quadratische, symmetrische Matrizen $\underline{\boldsymbol{A}}^\top = \underline{\boldsymbol{A}}$ definiert, wo gilt

$$\underline{\boldsymbol{A}} = \underline{\boldsymbol{Q}}\underline{\boldsymbol{D}}\underline{\boldsymbol{Q}}^\top,$$

wobei $\underline{\boldsymbol{Q}}^\top \underline{\boldsymbol{Q}} = \underline{\boldsymbol{I}}$ eine orthogonale Matrix mit Spalten $\boldsymbol{q}_i$ (Eigenvektoren) und $\underline{\boldsymbol{D}}$ eine Diagonalmatrix mit Diagonalelementen $d_i \in \mathbb{R}$ (Eigenwerten) ist. Es gilt also $\underline{\boldsymbol{A}}\boldsymbol{q}_i = d_i \boldsymbol{q}_i$.

Demgegenüber kann die SVD auf beliebige, auch nicht-quadratische $(m \times n)$-Matrizen $\underline{\boldsymbol{M}}$ angewendet werden. Die Zerlegung lautet

$$\underline{\boldsymbol{M}} = \underline{\boldsymbol{U}}\underline{\boldsymbol{\Sigma}}\underline{\boldsymbol{V}}^\top,$$

wobei $\underline{\boldsymbol{U}}^\top \underline{\boldsymbol{U}} = \underline{\boldsymbol{I}}_m$ eine orthogonale $(m \times m)$-Matrix und $\underline{\boldsymbol{V}}^\top \underline{\boldsymbol{V}} = \underline{\boldsymbol{I}}_n$ eine orthogonale $(n \times n)$-Matrix ist. Weiter ist $\underline{\boldsymbol{\Sigma}}$ eine $(m \times n)$-Diagonalmatrix mit *nichtnegativen* Diagonalelemeten $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_k \geq 0$, $k = \min\{m, n\}$. Die Diagonalelemente σ_i werden *Singulärwerte (SW)* der Matrix $\underline{\boldsymbol{M}}$ genannt. Weiter heissen die Spalten $\boldsymbol{u}_j$ von $\underline{\boldsymbol{U}}$ *Links-Singulärvektoren (Links-SV)* und die Spalten $\boldsymbol{v}_i$ von $\underline{\boldsymbol{V}}$ sind *Rechts-SV.* Für diese Grössen gilt

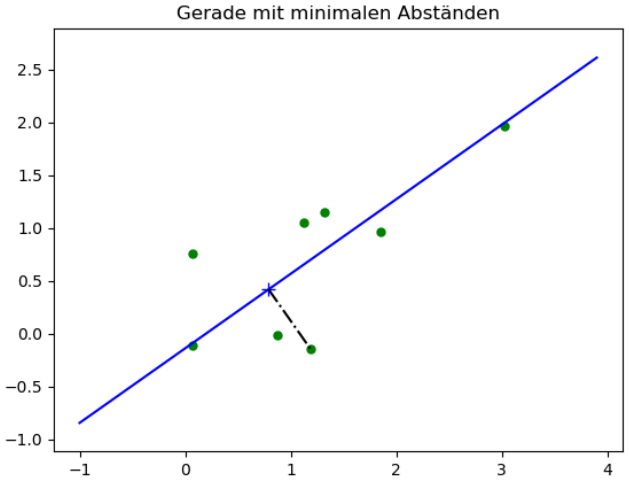
$$\underline{\boldsymbol{M}}\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i \quad \text{resp.} \quad \underline{\boldsymbol{M}}^\top \boldsymbol{u}_i = \sigma_i \boldsymbol{v}_i \quad \text{für} \quad i \in \{1, 2, \ldots, k\}.$$

Die letzten beiden Formeln zeigen die Verwandtschaft mit der Eigenwertzerlegung symmetrischer Matrizen.

Die numerische Berechnung der SVD ist aufwändig und anspruchsvoll. Dies überrascht wenig, wird doch auch die Numerik der Eigenwertzerlegung symmetrischer Matrizen im Mathematikunterricht an der FH nur am Rande thematisiert. Wir wollen uns im Rahmen dieses Projekts deshalb vornehmlich mit der *Interpretation* der SVD befassen. Es sind die damit verbundenen Einsichten, welche Ihnen in technischen Anwendungen helfen, die entsprechenden Methoden zu nutzen und zu beherrschen. Die eigentliche numerische Bestimmung von $\underline{\boldsymbol{U}}$, $\underline{\boldsymbol{V}}$ und $\underline{\boldsymbol{\Sigma}}$ überlassen wir einer Blackbox (MATLAB: `svd`, Python: `numpy.linalg.svd`). Konkrete Details dazu finden Sie auf den entsprechenden Hilfeseiten dieser Tools.

Aufgabe 1: Konsultieren Sie für weitere allgemeine Eigenschaften der SVD die angegebenen Quellen und notieren Sie die wichtigsten Punkte mit eigenen Worten zur späteren Referenz.

Geometrische Interpretation der SVD Das Bild unten zeigt $n = 8$ Punkte in der Ebene, gegeben durch deren Ortsvektoren $\boldsymbol{p}_i = (x_i, y_i)^\top$ sowie eine Gerade (Sie finden die Punktkoordinaten in der Datei `datenPunkte.csv`; zum Einlesen verwenden Sie mit Vorteil `readmatrix` in MATLAB, resp. `numpy.loadtxt` in Python). Die Gerade soll so bestimmt werden, dass die Summe der Abstandsquadrate minimal wird. Wichtig: der Abstand ist hier *geometrisch* definiert, d.h. er wird *rechtwinklig* zur Geraden gemessen und entspricht somit *nicht* der *vertikalen* Abweichung von der Geraden, wie bei der linearen Ausgleichsrechnung.



Wir gehen diese Problemstellung numerisch experimentell an: die Durchführung der unten beschriebenen Schritte führt Sie zur Bestimmung der gesuchten Geraden. Parallel dazu erarbeiten Sie sich eine geometrische Interpretation der SVD, die diesen Algorithmus nachvollziehbar macht. Ein formaler Beweis für die Korrektheit des Vorgehens ist jedoch nicht Ziel dieser Aufgabe.

- Im ersten Schritt verschiebt man den Koordinatenursprung in den Mittelpunkt der Punktwolke, definiert durch $\boldsymbol{p}_M = (x_M, y_M)^\top$ mit $x_M = \frac{1}{n}\sum_{i=1}^{n} x_i$ und $y_M = \frac{1}{n}\sum_{i=1}^{n} y_i$. Die neuen Ortsvektoren sind also $\boldsymbol{p}'_i = \boldsymbol{p}_i - \boldsymbol{p}_M$.
- Die Ortsvektorkomponenten werden in eine $(n\times 2)$-Matrix $\underline{\boldsymbol{Q}} = \begin{pmatrix} \boldsymbol{x}' & \boldsymbol{y}' \end{pmatrix}$ gepackt, mit $\boldsymbol{x}' = (x'_1, x'_2, \ldots, x'_n)^\top$ und $\boldsymbol{y}' = (y'_1, y'_2, \ldots, y'_n)^\top$.
- $\underline{\boldsymbol{Q}}$ wird der *ökonomischen* Variante der SVD unterzogen $\underline{\boldsymbol{Q}} = \underline{\boldsymbol{U}}\,\underline{\boldsymbol{\Sigma}}\,\underline{\boldsymbol{V}}^\top$, d.h. man erhält die $(n \times 2)$-Matrix $\underline{\boldsymbol{U}}$, die (2×2)-Matrix $\underline{\boldsymbol{\Sigma}}$ sowie die (2×2)-Matrix $\underline{\boldsymbol{V}}$ (zur Definition von "ökonomisch" konsultieren Sie SVD-Links oben).
- Nun der zentrale Schritt: Wir drücken die Zeilen der ursprünglichen Matrix $\underline{\boldsymbol{Q}}$ (also die Komponenten der Ortsvektoren) durch die Elemente von $\underline{\boldsymbol{U}}$, $\underline{\boldsymbol{\Sigma}}$ und $\underline{\boldsymbol{V}}$ aus. Idee: Jeder Ortsvektor wird als Linearkombination der Spaltenvektoren von $\underline{\boldsymbol{V}}$ (Zeilenvektoren von $\underline{\boldsymbol{V}}^\top$) aufgefasst. In den entsprechenden Formeln kommen sowohl Elemente von $\underline{\boldsymbol{U}}$ als auch die SW aus $\underline{\boldsymbol{\Sigma}}$ vor.
- Im Allgemeinen ist der erste SW viel grösser als der zweite (Frage: wie muss die Punktwolke beschaffen sein, damit dem so ist?). Unter dieser

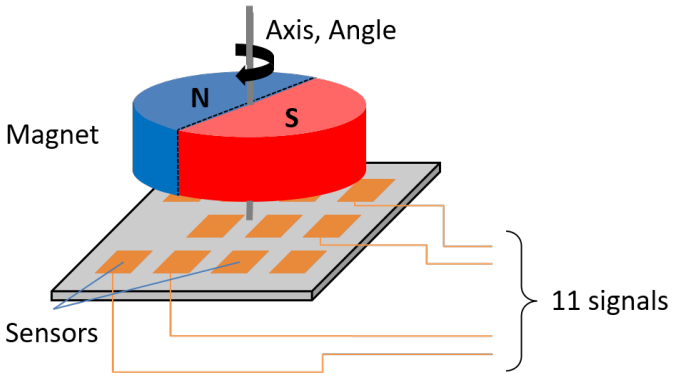
Voraussetzung ist folgende Approximation sinnvoll: In der Linearkombination für $\boldsymbol{p}'_i$ wird $\sigma_2 = 0$ gesetzt, was zu neuen Ortsvektoren $\tilde{\boldsymbol{p}}'_i$ führt. Behauptung: die Punkte $\tilde{\boldsymbol{p}}'_i$ liegen auf einer Graden und weisen minimalen geometrischen Abstand zu den Punkten $\boldsymbol{p}'_i$ auf (Wieso? Begründen Sie!).

- Schliesslich wird die Verschiebung des Ursprungs durch $\tilde{\boldsymbol{p}}_i = \tilde{\boldsymbol{p}}'_i + \boldsymbol{p}_M$ rückgängig gemacht. Behauptung: Die Punkte $\tilde{\boldsymbol{p}}_i$ liegen auf der gesuchten Geraden und haben den geforderten minimalen, rechtwinkligen Abstand von den Punkten $\boldsymbol{p}_i$!

Aufgabe 2: Führen Sie diese Schritte für die gegebenen Punkte durch, und stellen Sie die Resultate grafisch dar. Zum Vergleich soll auch die Ausgleichsgerade, welche die *vertikalen* Abstände minimiert, berechnet und dargestellt werden.

Wählen Sie aus den folgenden beiden Themen eines aus, mit dem Sie sich intensiver beschäftigen möchten. Bearbeiten Sie also nur *entweder* Aufgabe 3 *oder* Aufgabe 4.

Fehlerdetektor eines Winkelsensors



Die Abbildung oben zeigt einen Winkelsensor basierend auf einem drehbaren Permanentmagneten und einem Chip mit $n = 11$ integrierten Hallsensoren, welche verschiedene B-Feld-Komponenten an unterschiedlichen Positionen relativ zur Drehachse des Magneten messen.

Mit diesen Signalen kann die Position, also der Drehwinkel des Magneten, bestimmt werden. Im Prinzip würden jedoch 2 Signale zur Winkelberechnung genügen (etwa eines, das wie $\sin(\phi)$ und eines, das wie $\cos(\phi)$ variiert). Die vorliegende Redundanz wird genutzt, um eine zusätzliche Funktion des Sensors bereitzustellen, die so genannte *inhärente Fehlerdetektion.* Darunter versteht man das autonome Feststellen einer plötzlichen oder graduellen Verschlechterung der Messgenauigkeit, und zwar ohne die Notwendigkeit von Referenzmessungen!

Diese Funktion beruht auf der erwähnten Redundanz. Aufgrund der Vielfachmessungen sind die Signale *korreliert.* Diese Korrelation reagiert beispielsweise sehr empfindlich auf relative Lageänderungen von Magnet und Chip, etwa infolge eines mechanischen Schlages. Über die Zeit abwandernde Kennlinien der Hallelemente führen ebenfalls zu Veränderungen der Korrelationen. Mit einer inhärenten Fehlerdetektion können sichere Messsysteme realisiert werden, welche von sich aus "merken", wenn sie nicht mehr genau messen.

Die Signalkorrelation kann mithilfe der SVD aus Kalibrationsmessungen bei der Herstellung des Sensors bestimmt werden. Stellen Sie sich vor, dass diese Messsignale Punkte in einem 11-dimensionalen, abstrakten Vektorraum sind. Dabei "bevölkert" die Gesamtheit der denkbaren Signale nur einen *niedrigdimensionalen* Unterraum dieses 11-dimensionalen Vektorraums. Sobald sich dieser Unterraum gegebenüber seinem Anfangszustand (bei Kalibration) verändert, kann dies mithilfe der korrelierten Signale festgestellt werden.

Für diese Aufgabe erhalten Sie zwei Datensets `signalsCalibration.csv` und `signalsDrift.csv`. Erstere entsprechen den Signalen bei einer vollständigen Umdrehung des Magneten und wurden bei der Kalibration aufgenommen. Sie repräsentieren die fehlerfreie Sensorfunktion. Letzere sind Messungen von zufälligen Winkelstellungen, aufgenommen über einen längeren Zeitraum und weisen eine *Drift* auf, die mit der oben skizzierten Methode bestimmt werden soll. Das Vorgehen ist folgendermassen:

- Die 51 Kalibrationspositionen liefern je einen Signalvektor $\boldsymbol{q}_i \in \mathbb{R}^{11}$, $i = 1, \ldots 51$, womit die (51×11)-Matrix $\underline{\boldsymbol{Q}}$ gebildet wird. Nach der SVD von $\underline{\boldsymbol{Q}}$ kann mithilfe der SW die Dimension k des relevanten, d.h. "bevölkerten" Unterraums bestimmt werden.

  Konkret wird die Anzahl "grosser" SW gezählt. Vergessen Sie dabei nicht, die Signalpunktwolke im Vektorraum in den Ursprung zu verschieben, bevor Sie die SVD durchführen.

- Ausgehend von der Dimension k und der Matrix $\underline{\boldsymbol{V}}$ aus der SVD wird eine (11×11)-Projektionsmatrix $\underline{\boldsymbol{P}}$ bestimmt. Diese Matrix bewirkt eine *orthogonale* Projektion auf den relevanten Unterraum. Bilden Sie für diesen Schritt eine Analogie zur vorherigen Aufgabe, der Bestimmung der Geraden. Im Prinzip ist die vorliegende Fragestellung "nur" eine mehrdimensionale Verallgemeinerung dieses Vorgehens.

- Die Projektion auf das *orthogonale Komplemement* des Signalunterraums, d.h. die Matrix $\overline{\underline{\boldsymbol{P}}} = \underline{\boldsymbol{I}}_{11} - \underline{\boldsymbol{P}}$, bildet Messsignale $\boldsymbol{q}$ bei normaler Sensorfunktion auf den Nullvektor ab. Sobald das Sensorsystem jedoch verändert wurde, weist der Vektor $\overline{\boldsymbol{q}} = \overline{\underline{\boldsymbol{P}}}\boldsymbol{q}$ eine von Null verschiedene Länge auf. Mit andern Worten, die euklidische Norm $|\overline{\boldsymbol{q}}|$ kann als *Fehlerindikator* genutzt werden.

  Hinweis: Eigenschaften von Projektionsmatrizen finden Sie in Projektion (Lineare Algebra) oder Projection (linear algebra).

- Es bleibt nun nur noch die Auswertung des Fehlerindikators je für die Kalibrations- und die Driftmessungen. Aus der grafischen Inspektion dieser Grösse kann eine Fehlerschwelle definiert werden.

  Hinweis: wegen unvermeidlichem Rauschen liegen selbst die Kalibrationssignale nicht exakt im relevanten Unterraum.

Aufgabe 3: Setzen Sie diesen Fehlerdetektionsmechanismus um. Bestimmen Sie insbesondere eine *Fehlerschwelle* ε_{crit}, welche genaue Messungen $|\overline{\boldsymbol{q}}| \leq \varepsilon_{crit}$ von ungenauen $|\overline{\boldsymbol{q}}| > \varepsilon_{crit}$ trennt.

Gesichtserkennung in Graustufenbildern Eine weitere Anwendung der SVD ist die Gesichtserkennung, eine Variante von *supervised machine learning.*

Ausgangslage ist eine Sammlung von Beispielgesichtern. Sie finden diese im Anhang (37 Personen mit je 10 verschiedenen Gesichtsausdrücken der Grösse 56×68 Pixel, aus der *AT&T Database of Faces*). Mithilfe dieser Daten kann der zu entwickelnde Algorithmus "lernen", Gesichter zu erkennen.

Verwenden Sie Standardfunktionen `imread` in MATLAB oder `matplotlib.pyplot.imread` in Python, um Bilddateien als Matrizen einzulesen: jedes Graustufenpixel entspricht einem Matrixelement. Bevor diese Bildinformation mithilfe der SVD genutzt werden kann, müssen die Matrizen in Vektoren umgewandelt werden, indem z. B. deren Spalten hintereinander platziert werden und so einen Zeilenvektor mit $3808 = 56 \cdot 68$ Einträgen bilden.

Wir stellen uns vor, dass diese Beispielgesichter als Ortsvektoren in einem Vektorraum der Dimension 3808 vorliegen. Dort bevölkern diese Vektoren jedoch nur einen "kleinen" Unterraum (dazu wird wiederum den Mittelwert aller Beispielvektoren subtrahiert, um das Zentrum dieser Punktwolke in den Ursprung zu verschieben).

Zudem erwartet man, dass andere Gesichter, welche nicht Teil der Beispielmenge sind, ebenfalls "einen grossen Überlapp" mit diesem Unterraum aufweisen. Somit können wir Gesichtsbilder und Nicht-Gesichtsbilder anhand des "Überlapps mit dem Unterraum" der Beispielbilder unterscheiden. Die Methode zur Identifikation des relevanten Unterraums ist wiederum die SVD, nun angewendet auf eine sehr grosse Matrix.

Das Verzeichnis `Gesichter` enthält zwei zusätzliche Bilddateien. Mit dem Tulpenbild können Sie den typischen Überlappwert mit Bildern eruieren, welche *kein* Gesicht enthalten. Das Bild mit der Skyline (siehe oben) wird in der letzten Teilaufgabe genutzt, um die Position des Gesichts im Vordergrund zu finden.

Dazu wird aus der entsprechenden Matrix ein Teilbild der Grösse 56×68 entnommen und auf Überlapp mit dem Gesichterunterraum getestet. Wenn der Bildausschnitt Pixel für Pixel durch das Skylinebild verschoben wird, kann die Position des Gesichts automatisch bestimmt werden.

Vergleichen Sie diese Aufgabe mit dem Beispiel von vorher, der Fehlerdetektion. Sie werden unschwer die Parallelen erkennen und somit die Lösung von Aufgabe 3 so abwandeln, dass damit die Gesichtserkennung möglich wird. Bitte benutzen Sie für das Lesen und Darstellen von Bilddatein die Bordmittel von MATLAB (`double(imread(img))/255, imshow(img)`) oder Python (`plt.imread(img), plt.imshow(img, cmap='gray')`).

Aufgabe 4: Setzen Sie den Bilderkennungsalgorithmus um. Dazu können Sie sich an folgender Lösungsstruktur orientieren:

- Einlesen der Gesichtsbilddateien, umwandeln der Matrizen in Vektoren und Speichern dieser Daten als Zeilenvektoren in der Matrix $\underline{\boldsymbol{Q}}$. Sie sollen nur einen Teil der Gesichter tatsächlich nutzen, so dass Ihnen auch Testbilder zur Verfügung stehen, welche nicht Teil der Beispielmenge ("Trainingsdaten") sind.

- Anwendung der SVD zur Bestimmung des relevanten Unterraums. Eruieren Sie eine sinnvolle Grösse dieses Unterraumes, indem Sie experimentell nachprüfen, welche Unterraumdimension notwendig ist, um ein bekanntes oder unbekanntes Gesicht zu reproduzieren. Konkret projizieren Sie ein Bild auf den Gesichterunterraum. Diese Projektion (ein Vektor) kann wieder in eine Bildmatrix umgewandelt und graphisch dargestellt werden. Dokumentieren Sie das typische Verhalten in Abhängigkeit der Unterraumdimension.

- Für die Suche der Gesichtsposition im Skylinebild verwenden Sie folgende Grösse $\mathrm{ssd}(x) = |\boldsymbol{q}(x) - \underline{\boldsymbol{P}}\boldsymbol{q}(x)|^2$, wobei x die Position des Bildausschnitts, $\boldsymbol{q}$ ein Bildvektor und $\underline{\boldsymbol{P}}$ die Projektion auf den Gesichterunterraum ist. Die Bezeichnung ssd steht für "sum of squared distances" und ist ein Mass für den Unterschied des Bildes und dessen Projektion. Wenn Sie den Wert von $\mathrm{ssd}(x)$ für jeden Bildausschnitt bestimmen, dann können Sie das Skylinebild zusammen mit dem Funktionsgraphen von $\mathrm{ssd}(x)$ in zwei übereinanderliegenden `subplot`s darstellen und so diesen Suchalgorithmus und dessen Effizienz grafisch ins Bild setzen.

- Die Projektionsmatrix $\underline{\boldsymbol{P}}$ ist sehr gross. Entwerfen Sie den Algorithmus so, dass diese Matrix gar nicht gebildet werden muss, sondern lediglich der entsprechende Ausschnitt $\underline{\boldsymbol{V}}$ gespeichert und genutzt wird.