Datenstrukturen und Methoden zur Darstellung räumlicher akustischer Simulationen

Johann-Markus Batke
Hochschule Emden/Leer, Constantiaplatz 4, 26723 Emden, Deutschland
Email: johann-markus.batke@hs-emden-leer.de

Abstract

Ziel dieses Beitrags ist der Entwurf einer generischen Datenstruktur für akustische Simulationsumgebungen. Bedeutend dabei ist die Abstraktion der Software-Schnittstellen von Implementierungsproblemen und die Orientierung an der zu lösenden akustischen Aufgabe selbst. Der Schnittstellenentwurf der dargestellten Implementierungsvorschläge ist daher aus der mathematischen Darstellung der akustischen Probleme getrieben.

Einleitung

Räumliche akustische Simulationen lassen sich gut in numerischen Umgebungen wie Matlab/Octave, Scilab oder NumPy darstellen. Entsprechend existiert eine Vielzahl von Simulationsumgebungen und Toolboxen für etwa Raumakustik- oder Schallfeldsyntheseaufgaben, die teils generisch [1, 2], aber oft speziell auf das adressierte Problem zugeschnitten sind [3, 4, 5, 6, 7].

Der Unterschied der hier vorgestellten Simulationsumgebung zu vielen der existierenden Lösungen besteht in einem möglichst generischen Schnittstellenentwurf. Dieser Schnittstellenentwurf orientiert sich an der mathematischen Formulierung der akustischen Probleme und soll gleichzeitig auch eine einfache Visualisierung des Problems zulassen.

Beispielsweise wird der Druck eines Schallfelds abhängig vom Ort berechnet. Übergibt man der Routine zur Druckberechnung ein Objekt mit den Orts-Koordinaten einer Fläche, lässt sich unter Verwendung des gleichen Objekts auch eine Visualisierung der Druckverhältnisse einfach umsetzen.

Durch Objektorientierung wird eine Implementierung erreicht, die die Umsetzung der akustischen Probleme durch geeignete Datentypen umsetzt, deren Bearbeitung mit bekannten aus Matlab/Octave bereits bekannten Befehlen und Sprachkonstrukten ermöglicht.

Allen räumlichen akustischen Problemen gemeinsam, dass sie im Anschauungsraum $\mathbb{R}^3$ darstellbar sind. Dieser Beitrag hat daher das Ziel, neben der vorgestellten Simulationsumgebung prinzipiell geeignete Entwurfsmuster für räumliche akustische Programmier-Aufgaben herauszuarbeiten, die eine Interoperabilität verschiedener Simulationsumgebungen ermöglichen können.

Akustische Simulationsaufgaben

Anhand der Vorbetrachtung zweier typischer Simulationsaufgaben, der Berechnung einer ebenen Welle bzw. der Approximation eines Schallfelds, sollen die grundsätzlichen Erfordernisse für räumliche Darstellungen akustischer Zusammenhänge erörtert werden. Die abgeleiteten Anforderungen dienen als Ausgangspunkt für den Entwurf der hier dargestellten Datenstrukturen.

Beispiel Ebene Welle

Der Druck einer ebenen Welle wird dargestellt über [8]

$$p(\vec{r},\vec{k}) = \mathrm{e}^{\mathrm{i}\vec{r}^T\vec{k}}. \tag{1}$$

Dabei bezeichnet p den Druck an der Stelle $\vec{r}$, wobei sich die ebene Welle in Richtung des Wellenzahlvektors $\vec{k}$ ausbreitet. Zur Berechnung des inneren Produkts $\vec{r}^T\vec{k}$ ist die Darstelllung der Vektoren in kartesischen Koordinaten zweckmäßig.

Die Druckverteilung oft als Ebene im Raum, also für eine Vielzahl von Vektoren $\vec{r}$ dargestellt. Diese Menge an Vektoren kann bei der Implementierung etwa unter Octave oder Matlab in einer Matrix abgespeichert werden. Die Vektoren sind im Anschauungsraum 3-dimensional, damit ergibt sich bei L betrachteten Punkten eine $3 \times L$ -Matrix. Auch der Wellenzahlvektor kann für D verschiedende Richtungen und F Frequenzen variiert werden, was eine $3 \times D \times F$ -Matrix ergibt. Verfolgt man die Matrix-Darstellung aller beteiligten Größen weiter, ergibt sich für p eine $3 \times L \times D \times F$ -Matrix.

Beispiel Higher Order Ambisonics

Im Zusammenhang mit Higher Order Ambisonics (HOA) wird ein Schallfeld durch eine Reihenentwicklung approximiert [9]. Die verwendete Gleichung für den Druck des Schallfelds hat die Form

$$p(\vec{r},k) = \sum_{n\in N} \sum_{|m|\leq n} \hat{f}_n^m Y_n^m, \tag{2}$$

der Druck p hängt ab vom Koeffizienten $\hat{f}_n^m = f(k,r)$ und den Kugelflächenfunktionen $Y_n^m = f(\theta,\phi)$. Das Schallfeld wird also wieder für einen oder mehrere Punkte $\vec{r}$ im Raum betrachtet, wobei das Kugelkoordinatensystem verwendet wird. Die Beschreibung ganzer Flächen im Raum ist besonders effizient möglich, wenn p für Kombinationen aus einigen Werten für r, θ und ϕ berechnet wird. Die Wellenzahl k kommt als davon unabhängig als Parameter hinzu. Insgesamt wird für diesen Fall eine $3 \times R \times T \times P \times K$ -Matrix aufgespannt.

Anforderungen

Ausgehend von den beiden dargestellten akustischen Aufgaben ergeben sich folgende Anforderungen für räumliche Darstellungen:

1. Es werden Darstellungen von beliebig verteilten Punkten im Raum verwendet. Jeder Vektor muss damit einzeln beschrieben sein.
2. Die darzustellenden Punkte im Raum sind auf einem Raster geordnet. Es ist ausreichend, die Raumachsen des gewählten Koordinatensystems zu beschreiben.

Zunächst beliebig verteilte Punkte im Raum können weiteren Kriterien genügen, wie es etwa bei der Berechnung finiter Elementer der Fall ist [7]. Die Anordnungen von Punkten entsprechend einem Raster tritt in anderen numerischen Simulationsumgebungen auf [5, 6, 2].

Datenstrukturen und Methoden zur Darstellung

Als Programmierumgebung wurde in diesem Beitrag Octave/Matlab gewählt, da hier Matrix-Strukturen besonders gut unterstützt werden. Unabhängig davon ist die Einhaltung einiger Programmier-Paradigmen erstrebenswert, die nachfolgend kurz dargestellt werden. Danach folgen einige Beispiele zur vorliegenden Implementierung.

Paradigmen und Entwurfsziele

Ein sehr wichtiges Paradigma aus Sicht des Autors ist unter dem Kürzel „DRY“ bekannt [10]:

> Don’t repeat yourself.

Grundsätzlich hilft dieses Paradigma, die Wartbarkeit von Quelltexten zu erhöhen und die Gefahr logischer Widersprüche zu mindern. In Konsequenz sind folgende Punkte von Bedeutung:

- Kopieren, einfügen und minimal abändern von Quelltext („copy/paste“-Programmierung) ist immer zu vermeiden. Sobald ein Programmteil zweimal gebraucht wird, sollte man eine entsprechende Routine oder Funktion anlegen.
- Das Anlegen von mehreren Variablen prinzipiell gleichen Inhalts sollte vermieden werden. Statt zum Beispiel für zwei Frequenzen zwei Variablen

  ```
  x_f1 = 3;
  x_f2 = 4;
  ```

  anzulegen, wähl man ein Array mit Parameter

  ```
  x_f(1) = 3;
  x_f(2) = 4;
  ```

  Damit werden gleichartige Informationen auch logisch zusammen verwaltet.

Speziell für die Umsetzung in Octave/Matlab kommt ein weiteres Paradigma hinzu:

> Schleifen sind zu vermeiden.

Die Indizierung einzelner Werte innerhalb einer Matrix erfordert die Verwendung einer Schleife (wie z.B. der `for`-Schleife), was für Octave/Matlab eine verlangsamte Bearbeitung bedeutet. Die Verwendung von Schleifen lässt sich aber vermeiden:

- Die Verwendung der Vektor-Algebra, auch als „vectorization“ bezeichnet [11].
- Statt Indizes per Schleife zu iterieren, können auch logische Index-Operationen verwenden werden.

Für die bereits diskutierten akustischen Aufgabenstellungen ergibt sich damit die Darstellung skalarer Größen wie Druck oder Vektoren wie Ort $\vec{r}$ und Wellenzahlvektor $\vec{k}$ als mehrdimensionale Matrix.

Aufgrund der mathematischen Darstellung und der zumeist erfolgenden graphischen Darstellung von zum Beispiel Druckverteilungen von Schallfeldern im Raum ergibt sich die Notwendigkeit, das verwendete Koordinatensystem des Ortsvektors $\vec{r}$ in die jeweils benötigte Darstellung zu konvertieren.

Da das Koordinatensystem eine Eigenschaft der Raumdarstellung, also der Vektorinformation ist, ist eine objektbasierte Darstellung günstig. Darüber ist es möglich, einen Punkt im Raum darzustellen und zusätzlich dazu die Information zum verwendeten Koordinatensystem mitzuverwalten.

Datenstrukturen

Das dargestellte Entwurfsziel der Darstellung von Raumpunkten als Objekt macht die Definition von neuen Datentypen notwendig:

Anforderung 1, die Darstellung einer beliebigen Punktemenge im Raum, wird durch einen neuen Datentyp `vector` in Octave/Matlab umgesetzt (die Octave/Matlab-intern als `class` verwaltet werden).

Anforderung 2 betrifft ebenfalls die Darstellung einer Punktemenge, die in diesem Fall über die Achsenabschnitte des verwendeten Koordinatensystems umgesetzt wird. Da es sich hier eine Rasterung des Raums ergibt, heißt der neue Datentyp `vectorgrid`.

Für beiden neuen Datentypen sollen die in Octave/Matlab üblichen Methoden wie `size`, `display`, `numel` oder `length` zur Verfügung stehen. Für die räumliche grafische Darstellung stellt Octave/Matlab das `meshgrid` zusammen mit dem Befehl `surface` zur Verfügung. Diese Methoden werden entsprechend für das Vectorgrid umgesetzt.

Die dargestellte Implementierung ist als Toolbox **booast** veröffentlicht [12].

Vector

Der Datentyp `vector` beschreibt Punkte im Raum $\mathbb{R}^3$, die als Matrix geordnet gespeichert werden können.

Dieser Datentyp verlangt bei Definition nach der Angabe des Koordinatensystems, ebenso die Rückgabe des Variableninhalts durch die Methode `get()`. Eine ggf. notwendige Koordinatentransformation wird entsprechend berechnet.

Beispiel:

```
x = [1,0,0];
y = [0,1,0];
```

```
z = [0,0,1];
v = vector(x, y, z, 'cartesian');
get(v, 'spherical')

ans =

   1.00000   1.00000   1.00000
   1.57080   1.57080   0.00000
   0.00000   1.57080   0.00000
```

Aufgrund der Verwendung im akustischen Kontext stehen als zusätzliche Methoden `normalize()` zur Normierung der Vektorlänge oder `scale()` zur Skalierung der Vektorlänge zur Verfügung.

Vectorgrid

Der Datentyp `vectorgrid` ist durch die Achseneinteilung des Koordinatensystems in einem betrachteten Raum definiert. Werden alle Kombinationen der Achseinteilung benötigt, so lässt sich das `vectorgrid` in eine Matrix von Vektoren vom Typ `vector` verwandeln.

Bespiel:

```
vg = vectorgrid(linspace(0, 10, 10), ...
                linspace(0, 10, 20), ...
                linspace(0, 10, 30), ...
                'cartesian');
v = vector(vg);
whos

Variables in the current scope:

 Attr Name        Size              Bytes  Class
 ==== ====        ====              =====  =====
      ans         1x1                   8  double
      v           10x20x30         144049  vector
      vg          10x20x30            497  vectorgrid

Total is 3 elements using 144554 bytes
```

Damit ist eine kompakte Darstellung des Anschauungsraums in *einer* Variable möglich, die sich über Indizierung (das sog. „slicing“) auch flexibel handhaben lässt. Die graphische Darstellung erfolgt wie beim bekannten `meshgrid` z.B. über den Befehl `surface`.

Anwendungsbeispiel

Einige akustische Aufgabenstellungen liegen in der Toolbox **booast** bei, etwa die Berechnung des Drucks ebener Wellen.

Die Darstellung einer ebenen Welle soll als Anwendungsbeispiel zeigen, das mit den an das akustische Problem angepassten Datentypen eine sehr kompakte Beschreibung der Simulationsaufgabe möglich ist.

Zunächst wird der Anschauungsraum als `vectorgrid` beschrieben:

```
x_A = linspace(0, 10, 60);
y_B = linspace(0, 10, 30);
z_C = linspace(0, 10, 40);
g_ABC = vectorgrid(x_A, y_B, z_C, ...
                   'cartesian');
```

Für die grafische Darstellung werden einige Raumebenen ausgewählt:

```
g_AB1 = g_ABC(:,:,1);
g_A1C = g_ABC(:,20,:);
g_1BC = g_ABC(1,:,:);
```

Nur für diese Auswahl wird nun der Druck der ebenen Welle berechnet, wozu das `vectorgrid` in den Typ `vector` umgesetzt wird:

```
%% direction/wavenumbervector of planewave
v = vector([1, 0.1, 0.1]', 'cartesian');
p_AB1 = planewave(vector(g_AB1), v);
p_A1C = planewave(vector(g_A1C), v);
p_1BC = planewave(vector(g_1BC), v);
```

Schließlich wird das Ergebnis visualisiert:

```
surface(g_AB1, real(p_AB1));
surface(g_A1C, real(p_A1C));
surface(g_1BC, real(p_1BC));
```

Es ergibt sich Abbildung 1. Nur durch geänderte Definition des Anschauungsraums in Kugelkoordinaten, wie etwa

```
A = 30; B = 40; C = 50;
r_A     = linspace(0, 10, A);
theta_B = linspace(0, pi, B);
phi_C   = linspace(0, 1.5*pi, C);
g_ABC   = vectorgrid(r_A, theta_B, phi_C, ...
                     'spherical');
```

und entsprechender Auswahl der dargestellten Ebenen

```
g_AB1 = g_ABC(:,:,10);  % circle
g_A1C = g_ABC(:,10,:);  % conus
g_1BC = g_ABC(20,:,:);  % sphere
```

ergibt sich Abbildung 2. Die Umrechnung zur Darstellung in kartesische Koordinaten ist für den Nutzer verborgen.

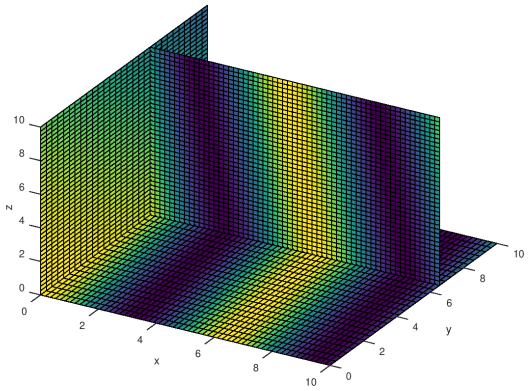

Abbildung 1: Eine ebene Welle, berechnet in drei Ebenen im Raum. Die darstellten Ebenen werden im Programmtext als `vectorgrid` darstellt. Die Berechnung des Drucks erfolgt nach temporärer Umwandlung in den Datentyp `vector`.

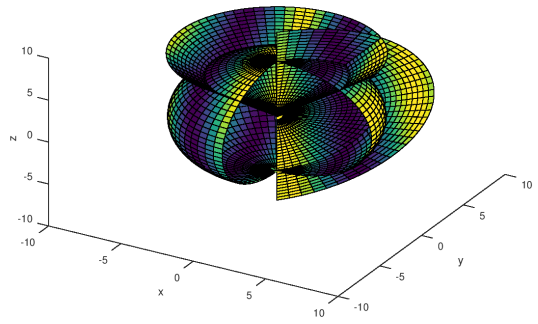

Abbildung 2: Eine ebene Welle, berechnet in drei Ebenen des Kugelkoordinatensystems. Die Schnittstellen für Berechnung und Darstellung sind in der Toolbox **booast** gleichartig.

Zusammenfassung

Ausgehend von typischen Simulationsaufgaben wie der Berechnung des Drucks im Raum durch eine ebene Welle oder der Approximation eines Schallfelds durch Higher Order Ambisonics wurden grundsätzliche Erfordernisse an die Darstellung des benötigten Ortsvektors $\vec{r}$ zusammengefasst.

In diesen Kontext zeigt sich die Verwendung von beliebig verteilbaren Punktemengen als Typ `vector` oder als über die Raumkoordinaten geordnetes Raster `vectorgrid` als ausreichend. Eine beispielhafte objektorientierte Implementierung als Toolbox **booast** sowie deren Anwendung wurde vorgestellt.

Literatur

[1] Marco Berzborn u. a. “The ITA-Toolbox: An Open Source MATLAB Toolbox for Acoustic Measurements and Signal Processing”. In: *43th Annual German Congress on Acoustics, Kiel (Germany), 6 Mar 2017 - 9 Mar 2017*. März 2017. URL: `http://publications.rwth-aachen.de/record/687308`.

[2] Hagen Wierstorf und Sascha Spors. “Sound Field Synthesis Toolbox”. In: *132nd Convention of the Audio Engineering Society*. 2012.

[3] A. Heller und E. M. Benjamin. “The Ambisonic Decoder Toolbox: Extensions for Partial-Coverage Loudspeaker Arrays”. In: *Linux Audio Conference 2014*. (Karlsruhe). 2014.

[4] Fabian Brinkmann und Stefan Weinzierl. “AKtools — An Open Software Toolbox for Signal Acquisition, Processing, and Inspection in Acoustics”. In: *Audio Engineering Society Convention 142*. Mai 2017. URL: `http://www.aes.org/e-lib/browse.cfm?elib=18685`.

[5] Bradley E. Treeby und B. T. Cox. “k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields”. In: *Journal of Biomedical Optics* 15.2 (2010).

[6] Christian Hoene, Isabel C. Patino Mejia und Alexandru Cacerovschi. “MySofa—Design Your Personal HRTF”. In: *Audio Engineering Society Convention 142*. Mai 2017. URL: `http://www.aes.org/e-lib/browse.cfm?elib=18640`.

[7] Qianqian Fang und David Boas. “Tetrahedral mesh generation from volumetric binary and gray-scale images”. In: *Proceedings of IEEE International Symposium on Biomedical Imaging 2009*. 2009, S. 1142–1145.

[8] Heinrich Kuttruff. *Room Acoustics*. fifth edition. Focal, 2009.

[9] Earl G. Williams. *Fourier Acoustics*. Bd. 93. Applied Mathematical Sciences. Academic Press, 1999.

[10] Wyatt Matthews. *Dont Repeat Yourself*. 2014. URL: `http://wiki.c2.com/?DontRepeatYourself`.

[11] Mathworks Matlab Documentation, Hrsg. *Vectorization*. Version R2018a. 2018. URL: `https://de.mathworks.com/help/matlab/matlab_prog/vectorization.html`.

[12] Johann-Markus Batke. *Bad Object Oriented Acoustics Simulation Toolbox*. 2018. URL: `https://github.com/badkey/booast`.