Discrete Differential Geometry: An Applied Introduction
Mark Gillespie • CMU 15-458/858B
Outline

• Brief overview of ddg-exercise-js
• Halfedge data structure
• Sparse matrices
• Solving linear systems (direct methods)
ddg-exercises-js
ddg-exercises-js

- Repository on Github
- https://github.com/cmu-geometry/ddg-exercises-js
- Contains all assignments for the semester
Javascrip

- Feels similar to C, C++, Java, …. Really any language with braces
- Runs in your browser, so there isn’t too much setup
- You probably won’t need to use any fancy features particular to Javascript - just need some functions, conditionals, loops, etc
ddg-exercises-js

• Documentation included
  ddg-exercises-js/docs/index.html

• Coding assignments
  ddg-exercises-js/projects

• Tests
  ddg-exercises-js/tests
ddg-exercises-js

ddg-exercises-js is a fast and flexible framework for 3D geometry processing on the web! Easy integration with HTML/WebGL makes it particularly suitable for things like mobile apps, online demos, and course content. For many tasks, performance comes within striking distance of native (C++) code. Plus, since the framework is pure JavaScript, no compilation or installation is necessary on any platform. Moreover, geometry processing algorithms can be edited in the browser (using for instance the JavaScript Console in Chrome).

At a high level, the framework is divided into three parts - an implementation of a halfedge mesh data structure, an optimized linear algebra package and skeleton code for various geometry processing algorithms. Each algorithm comes with its own viewer for rendering.

Detailed documentation and unit tests for each of these parts can be found in the docs and tests directories of this repository.

Getting started

1. Clone the repository and change into the projects directory

   ```bash
   git clone https://github.com/cmu-geometry/ddg-exercises-js.git
   cd ddg-exercises-js/projects
   ```

2. Open the index.html file in any of the sub-directories in a browser of your choice (Chrome and Firefox usually provide better rendering performance than Safari).

Dependencies (all included)

1. Linear Algebra - A wrapper around the C++ library Eigen compiled to work well with WebAssembly.
Coding Assignments

• Viewers

ddg-exercises-js/projects/simplicial-complex-operators/index.html

• Write code in project folder or one of the modules

• Graphics programming often involves a lot of boilerplate before getting started drawing - We’ve mostly done that for you. You just have to fill in the interesting bits
Tests

• Test scripts
ddg-exercises-js/tests/simplicial-complex-operators/test.html

• As you write your code, you should see it pass more tests
The Halfedge Data Structure
The Halfedge Data Structure

```c
struct Halfedge {
    Halfedge twin;
    Halfedge next;
    Vertex vertex;
    Edge edge;
    Face face;
};

struct Edge {
    Halfedge halfedge;
};

struct Face {
    Halfedge halfedge;
};

struct Vertex {
    Halfedge halfedge;
};
```
The Halfedge Data Structure

How would I find the faces adjacent to an edge?

Given: `Edge e`

```c
struct Vertex {
    Halfedge halfedge;
};

struct Edge {
    Halfedge halfedge;
};

struct Face {
    Halfedge halfedge;
};

struct Halfedge {
    Halfedge twin;
    Halfedge next;
    Vertex vertex;
    Edge edge;
    Face face;
};
```
The Halfedge Data Structure

How would I find the faces adjacent to an edge?

Given: `Edge e`

```c
Halfedge he = e.halfedge;
Face left_face  = he.face;
Face right_face = he.twin.face;
```
The Halfedge Data Structure

How would I find the edges adjacent to a triangle?

Given: `Face tri`

```c
struct Vertex {
    Halfedge halfedge;
};

struct Edge {
    Halfedge halfedge;
};

struct Face {
    Halfedge halfedge;
};

struct Halfedge {
    Halfedge twin;
    Halfedge next;
    Vertex vertex;
    Edge edge;
    Face face;
};
```
The Halfedge Data Structure

How would I find the edges adjacent to a triangle?

Given: `Face tri`

```plaintext
Halfedge he = tri.halfedge;
Edge e1 = he.edge;
Edge e2 = he.next.edge;
Edge e3 = he.next.next.edge;
```
The Halfedge Data Structure

How would I loop over the edges adjacent to a polygon?
Given: \texttt{Face f}

```cpp
struct Vertex {
    Halfedge halfedge;
};

struct Edge {
    Halfedge halfedge;
};

struct Face {
    Halfedge halfedge;
};

struct Halfedge {
    Halfedge twin;
    Halfedge next;
    Vertex vertex;
    Edge edge;
    Face face;
};
```

```plaintext
f.halfedge
```
How would I loop over the edges adjacent to a polygon? Given: Face `f`

```
Halfedge start = f.halfedge;
Halfedge he = start;
do {
    Edge e = he.edge;
    /* Some code */
    he = he.next;
} while (he != start);
```
The Halfedge Data Structure

How would I loop over the edges adjacent to a vertex?

Given: \texttt{Vertex } v

\hspace{1cm} v.halfedge
The Halfedge Data Structure

How would I loop over the edges adjacent to a vertex?

Given: `Vertex v`

```c
Halfedge start = v.halfedge;
Halfedge he = start;
do {
    Edge e = he.edge;
    /* Some code */
    he = he.twin.next;
} while (he != start);
```
In ddg-exercises-js

geometry-processing.js  Modules  Classes  Global  Search

Class: Mesh
Core. Mesh

new Mesh()
This class represents a Mesh.

Properties:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertices</td>
<td>Array.&lt;module/Core/Vertex&gt;</td>
<td>The vertices contained in this mesh.</td>
</tr>
<tr>
<td>edges</td>
<td>Array.&lt;module/Core/Edge&gt;</td>
<td>The edges contained in this mesh.</td>
</tr>
<tr>
<td>faces</td>
<td>Array.&lt;module/Core/Face&gt;</td>
<td>The faces contained in this mesh.</td>
</tr>
<tr>
<td>corners</td>
<td>Array.&lt;module/Core/Corner&gt;</td>
<td>The corners contained in this mesh.</td>
</tr>
<tr>
<td>halfedges</td>
<td>Array.&lt;module/Core/Halfedge&gt;</td>
<td>The halfedges contained in this mesh.</td>
</tr>
<tr>
<td>boundaries</td>
<td>Array.&lt;module/Core/Face&gt;</td>
<td>The boundary loops contained in this mesh.</td>
</tr>
<tr>
<td>generators</td>
<td>Array.&lt;module/Core/Halfedge&gt;</td>
<td>An array of halfedge arrays, i.e., [[h11, h21, ..., h1n], [h12, h22, ..., h2n], ...] representing this mesh's homology generators.</td>
</tr>
</tbody>
</table>

Methods

geometry-processing.js  Modules  Classes  Global  Search

Class: Vertex
Core. Vertex

struct Vertex

new Vertex()
This class represents a vertex in a Mesh.

Properties:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>halfedges</td>
<td>module/Core/Halfedge</td>
<td>One of the outgoing halfedges associated with this vertex.</td>
</tr>
</tbody>
</table>

geometry-processing.js  Modules  Classes  Global  Search

Class: Edge
Core. Edge

struct Edge

new Edge()
This class represents an edge in a Mesh.

Properties:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>halfedges</td>
<td>module/Core/Halfedge</td>
<td>One of the halfedges associated with this edge.</td>
</tr>
</tbody>
</table>

geometry-processing.js  Modules  Classes  Global  Search

Class: Face
Core. Face

struct Face

new Face()
This class represents a face in a Mesh.

Properties:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>halfedges</td>
<td>module/Core/Halfedge</td>
<td>One of the halfedges associated with this face.</td>
</tr>
</tbody>
</table>
In ddg-exercises-js

Includes many convenience functions

**adjacentVertices()**

Convenience function to iterate over the vertices in this face. Iterates over the vertices of a boundary loop if this face is a boundary loop.

**Example**

```javascript
let f = mesh.faces[1]; // or let b = mesh.boundaries[1]
for (let v of f.adjacentVertices()) {
    // Do something with v
}
```

**adjacentEdges()**

Convenience function to iterate over the edges adjacent to this vertex.

**Example**

```javascript
let v = mesh.vertices[0];
for (let e of v.adjacentEdges()) {
    // Do something with e
}
```
Storing Matrices
Matrices

How can I write down a matrix?

• Option 2: 2D array

• If your matrix doesn’t have much structure, this might be the best you can do

• But it can take a lot of space to write down an entire matrix

• And working with (really) big matrices is slow
Matrices

- What matrices do we care about?

- It turns out that *adjacency matrices* are very important

$$E^0 = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \\ 4 & 0 & 1 & 0 \\ 5 & 0 & 0 & 1 \end{bmatrix}$$

$$E^1 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$
Matrices

• Most entries are 0!

• We can improve our lives by only storing nonzero entries

• Sparse matrix

Class: SparseMatrix

LinearAlgebra. SparseMatrix

new SparseMatrix()

This class represents a m by n real matrix where only nonzero entries are stored explicitly. Do not create a SparseMatrix from its constructor, instead use static factory methods such as fromTriplet, identity and diag.

Example

```javascript
let T = new Triplet(100, 100);
T.addEntry(3.4, 11, 43);
T.addEntry(5.4, 99, 99);
let A = SparseMatrix.fromTriplet(T);
let B = SparseMatrix.identity(10, 10);
let d = DenseMatrix.ones(100, 1);`
Aside: Sparse Matrix Formats

• Important format: Compressed Sparse Row (CSR)

• Store the nonzero entries in row-major order, and some information about spacing

• Row-major order => matrix-vector products are fast

\[
A[i] = \text{entries} \\
IA[i] = \text{total number of nonzero entries before row } i \\
JA[i] = \text{column of the } i\text{th entry of } A
\]
Aside: Sparse Matrix Formats

\[ A[i] = \text{entries} \]
\[ IA[i] = \text{total number of nonzero entries before row } i \]
\[ JA[i] = \text{column of the } i\text{th entry of } A \]

\[
\begin{pmatrix}
0 & 0 & 0 & 0 \\
5 & 8 & 0 & 0 \\
0 & 0 & 3 & 0 \\
0 & 6 & 0 & 0
\end{pmatrix}
\]

\[ A = \begin{bmatrix} 5 & 8 & 3 & 6 \end{bmatrix} \]
\[ IA = \begin{bmatrix} 0 & 0 & 2 & 3 & 4 \end{bmatrix} \]
\[ JA = \begin{bmatrix} 0 & 1 & 2 & 1 \end{bmatrix} \]
Aside: Sparse Matrix Formats

• There's also Compressed Sparse Column (CSC)

• Fast to multiply CSC by row vectors

• Both are slow to add elements to
  • Usually you build the matrix in another format, then convert before doing computation
Sparse Matrices in `ddg-exercises-js`

- Build from Triplet
- Modified version of CSC/CSR
- Eigen

**Class: SparseMatrix**

```javascript
new SparseMatrix()
```

This class represents an m by n real matrix where only nonzero entries are stored explicitly. Do not create a SparseMatrix from its constructor; instead use static factory methods such as fromTriplet, identity and diag.

**Example**

```javascript
let T = new Triplet(100, 100);
T.addEntry(3, 4, 12);
let A = SparseMatrix.fromTriplet(T);
let B = SparseMatrix.identity(10, 10);
let d = DenseMatrix.ones(100, 10);
let C = SparseMatrix.diag(d);
```

**Class: Triplet**

```javascript
new Triplet(m, n)
```

This class represents a small structure to hold nonzero entries in a SparseMatrix. Each entry is a triplet of a value and the (i, j)th indices, i.e., (a, i, j).

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>module LinearAlgebra.Triplet</td>
<td>A triplet object containing only the nonzero entries that need to be stored in this sparse matrix.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>number</td>
<td>The number of rows in the sparse matrix that will be initialized from this triplet.</td>
</tr>
<tr>
<td>n</td>
<td>number</td>
<td>The number of columns in the sparse matrix that will be initialized from this triplet.</td>
</tr>
</tbody>
</table>
Warning

• How do you represent a vector?
• LinearAlgebra.Vector only represents 3D vectors
• Instead, construct a matrix with n rows and 1 column
• Multiply matrices by vectors using timesDense or timesSparse
Linear Systems of Equations

Linear algebra review

\[
\begin{align*}
x + 2y - 4z &= 1 \\
3x - 5y + 7z &= 2 \\
-x + 3y + 5z &= -2
\end{align*}
\]

\[
\begin{pmatrix}
1 & 2 & -4 \\
3 & -5 & 7 \\
-1 & 3 & 5
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
= \begin{pmatrix}
1 \\
2 \\
-2
\end{pmatrix}
\]

\[A \vec{x} = \vec{b}\]
Linear Systems of Equations

- How do we solve $Ax = b$?
- Compute the inverse / Gaussian Elimination
- Not good for sparse matrices
Linear Systems of Equations

• Some special cases are easy

• What if $A$ is diagonal?

\[
\begin{pmatrix}
1 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & -3
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
=
\begin{pmatrix}
1 \\
4 \\
6
\end{pmatrix}
\]
Linear Systems of Equations

• What if $A$ is lower-triangular?

\[
\begin{pmatrix}
1 & 0 & 0 \\
1 & 2 & 0 \\
2 & 3 & -3
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
= 
\begin{pmatrix}
1 \\
5 \\
11
\end{pmatrix}
\]

\[
x = 1 \\
x + 2y = 5 \implies y = 2 \\
2x + 3y - 3z = 11 \implies z = -1
\]

• (Same trick works if $A$ is upper-triangular)
Linear Systems of Equations

• Can this help us with arbitrary linear systems?

• Yes!

• Given an invertible matrix $A$, we can factor it as a lower-triangular matrix times an upper triangular matrix:

$$A = LU$$

$$\begin{pmatrix} 4 & 3 \\ 6 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1.5 & 1 \end{pmatrix} \begin{pmatrix} 4 & 3 \\ 0 & -1.5 \end{pmatrix}$$
LU Decomposition

\[ Ax = b \]

\[ LUx = b \]

\[ Ly = b \quad \text{and} \quad y = Ux \]
LU Decomposition

• How do we compute LU decomposition?
• Simple solution - run Gaussian Elimination half way
  • Problem - still not good for sparse matrices
• We’ll use a fancier implementation

```
lu()

Returns a sparse LU factorization of this sparse matrix.
```

Class: LU

LinearAlgebra. LU

new LU()

This class represents a LU factorization of a square SparseMatrix. The factorization is computed on the first call to solveSquare, and is reused in subsequent calls to solveSquare (e.g. when only the right hand side b of the linear system Ax = b changes) unless the sparse matrix itself is altered through operations such as *=, += and -=. Do not use the constructor to initialize this class, instead access the LU factorization of a sparse matrix directly from the matrix itself.

Example

```
// solve the linear system Ax = b, where A is a square sparse matrix
let A = SparseMatrix.Identity(5, 5);
let b = DenseMatrix.ones(5, 1);

let lu = A.lu();
let x = lu.solveSquare(b);

b.scaleBy(-1);
x = lu.solveSquare(b); // factorization is reused
```

Methods

solveSquare(b)

Solves the linear system Ax = b, where A is a square sparse matrix.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>module:LinearAlgebra.DenseMatrix</td>
<td>The dense right hand side of the linear system Ax= b.</td>
</tr>
</tbody>
</table>
Cholesky Decomposition

• If A is symmetric and positive-semidefinite, then the LU decomposition is really nice

\[ A = LL^T \]

• Called Cholesky or LLT decomposition
QR Decomposition

• LU and Cholesky decompositions take advantage of the fact that it’s easy to solve triangular systems

• It’s also easy to solve systems given by rotation matrices

\[ Q^{-1} = Q^T \]

\[ Qx = b \implies x = Q^T b \]
QR Decomposition

• Any square matrix can be decomposed as QR for Q a rotation and R upper triangular

• There are also versions for rectangular matrices

\[ Ax = b \]

\[ QRx = b \]

\[ Qy = b \quad \text{and} \quad y = Rx \]
QR Decomposition

- Also available in framework
- Not as fast as Cholesky but more widely applicable

```sql
qr()

Returns a sparse QR factorization of this sparse matrix.

Type
module:LinearAlgebra.QR

solve(b)

Solves the linear system Ax = b, where A is a rectangular sparse matrix.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>module:LinearAlgebra.DenseMatrix</td>
<td>The dense right hand side of the linear system Ax = b</td>
</tr>
</tbody>
</table>
```
Discrete Differential Geometry: An Applied Introduction
Mark Gillespie • CMU 15-458/858B

Thanks!