The STCE Scripting Language

Support for Linear Algebra

Uwe Naumann

Informatik 12 (STCE), RWTH Aachen University

Contents

Outline

Eigen

Vectors

Matrices

Linear Algebra
    Arithmetic
    Solution of Systems of Linear Equations

Vector and Matrix Types

```
#pragma once

#include ”Eigen/Dense”

namespace Eigen {

template<typename T, int N=Dynamic>
using vector_t=Matrix<T,N,1>;

template<typename T, int M=Dynamic, int N=M>
using matrix_t=Matrix<T,M,N>;

}
```

- Base-type-generic dynamically sized vector and matrix types are provided by Eigen, a C++ template library for linear algebra.
- Two predominantly used specializations are defined in the header file Eigen.hpp, which is part of the sample code.

- `#pragma once` avoids multiple inclusion of the same header file.
- Conceptually, the entire range of functionalities provided by Eigen is available for STCE scripting; see `https://eigen.tuxfamily.org/`.

Outline

Eigen

Vectors

Matrices

Linear Algebra
    Arithmetic
    Solution of Systems of Linear Equations

Vectors

```
1  #include<iostream>
2  using namespace std;
3
4  #include ”Eigen.hpp”
5
6  int main() {
7    Eigen::vector_t<float> v(3);
8    v(0)=1e−1; v(2)=1e−3;
9    cout << v << endl;
10   return 0;
11 }
```

yields output

```
  0.1
    0 // not guaranteed
0.001
```

- The base-type-generic dynamically sized vector type `Eigen::vector_t<T>` enables vector arithmetic.
- Uninitialized vectors are allocated dynamically (line 6).
- Access to individual entries requires specification of the index in parentheses (line 7).

Vector Constants

```
1  #include<iostream>
2  using namespace std;
3
4  #include ”Eigen.hpp”
5
6  int main() {
7    using T=Eigen::vector_t<float>;
8    T v=T::Zero(3); cout << v.transpose() << endl;
9    v=T::Ones(3); cout << v.transpose() << endl;
10   v=T::Unit(3,1); cout << v.transpose() << endl;
11   v=T::Random(3); cout << v.transpose() << endl;
12   return 0;
13 }
```

- A *static member* function f of a *class* T is called as T::f; e.g. initialization with (e.g. second) Cartesian basis (also: unit) vector (e.g. in R^3) implemented as static member function of vector_t<T> (line 10).

yields output

```
0 0 0
1 1 1
0 1 0
 0.680375 −0.211234 0.566198
```

- A *non-static member* function f is called on an *object* v of type T as v::f; e.g. the transpose is implemented as a non-static member function of Eigen::vector_t<T> (line 8).

Outline

Eigen

Vectors

Matrices

Linear Algebra
  - Arithmetic
  - Solution of Systems of Linear Equations

Matrices

```
1  #include<iostream>
2  using namespace std;
3
4  #include "Eigen.hpp"
5
6  int main() {
7    Eigen::matrix_t<float> m(2,3);
8    m(0,0)=1e−38; m(1,2)=1e2;
9    cout << m << endl;
10   return 0;
11 }
```

yields output

```
1  1e−38 0 0
2        0 0 100
3  // zeros are not guaranteed
```

- The base-type-generic dynamically sized matrix type `Eigen::matrix_t<T>` enables matrix and matrix-vector arithmetic.
- Uninitialized matrices are allocated dynamically (line 7).
- Access to individual entries requires specification of the row and column indexes in parentheses (line 8).

Matrix Constants

```
1  #include<iostream>
2  using namespace std;
3
4  #include "Eigen.hpp"
5
6  int main() {
7    using T=Eigen::matrix_t<float>;
8    T m=T::Zero(2,2); cout << m << endl;
9    m=T::Ones(2,2); cout << m << endl;
10   m=T::Identity(2,2); cout << m << endl;
11   m=T::Random(2,2); cout << m << endl;
12   return 0;
13 }
```

yields output

```
0 0
0 0
1 1
1 1
1 0
0 1
 0.680375 0.566198
-0.211234 0.59688
```

- Eigen::matrix_t<T> constants are similar to Eigen::vector_t<T> constants.
- E.g. the identity (e.g. in R^2) is implemented as a static member function of Eigen::matrix_t<T> (line 10).

Outline

Eigen

Vectors

Matrices

Linear Algebra

- Arithmetic
- Solution of Systems of Linear Equations

Inner Vector Product

```
1  #include <iostream>
2  using namespace std;
3
4  #include "Eigen.hpp"
5
6  int main() {
7    int n=3;
8    using T=Eigen::vector_t<float>;
9    T v=T::Random(n);
10   cout << v.transpose() << endl;
11   float vTv=v.dot(v);
12   cout << vTv << "="
13        << v.squaredNorm() << endl;
14   return 0;
15 }
```

yields output

```
 0.680375 −0.211234 0.566198
0.828111=0.828111
```

- The inner (dot) product of two vectors $u, v \in \boldsymbol{R}^n$ is defined as

$$u^T \cdot v = \sum_{i=0}^{n-1} u_i \cdot v_i \in \boldsymbol{R} \; .$$

- `Eigen::vector_t` provides the non-static member function `dot` (line 11).
- Note that

$$v^T \cdot v = \|v\|_2^2 = \sum_{i=0}^{n-1} v_i^2$$

(lines 12-13).

Matrix-Vector Product

```
1   #include <iostream>
2   using namespace std;
3
4   #include "Eigen.hpp"
5
6   int main() {
7     int m=2,n=3;
8     using VT=Eigen::vector_t<float>;
9     using MT=Eigen::matrix_t<float
        >;
10    VT x=VT::Random(n);
11    MT A=MT::Random(m,n);
12    VT y=A*x;
13    cout << y.transpose() << endl;
14    return 0;
15  }
```

yields output

```
0.83762 0.378115
```

- The product of a matrix $A(A_{j,i}) \in \boldsymbol{R}^{m\times n}$ with a vector $x = (x_i) \in \boldsymbol{R}^n$ is a vector $y = A \cdot x \in \boldsymbol{R}^m$ defined as

$$y = (y_j) \equiv \left(\sum_{i=0}^{n-1} A_{j,i} \cdot x_i\right)_{j=0,\ldots,m-1} .$$

- The operator $*$ is overloaded accordingly (line 12).

Matrix-Matrix Product

```
1  #include <iostream>
2  using namespace std;
3
4  #include "Eigen.hpp"
5
6  int main() {
7    int m=2,n=3;
8    using T=Eigen::matrix_t<float>;
9    T A=T::Random(m,n);
10   T B=T::Random(n,m);
11   T C=A*B;
12   cout << C << endl;
13   return 0;
14 }
```

yields output

```
-0.286392 0.260042
 0.658662 -0.20569
```

- The product of two matrices $A \in \boldsymbol{R}^{m \times n}$ and $B \in \boldsymbol{R}^{n \times p}$ is a matrix $C = A \cdot B \in \boldsymbol{R}^{m \times p}$ defined as

$$C = (C_{k,i}) \equiv \left(\sum_{j=0}^{n-1} A_{k,j} \cdot B_{j,i} \right)_{i=0,\ldots,p-1}^{k=0,\ldots,m-1} .$$

- The operator $*$ is overloaded accordingly (line 11).

Solution of Systems of Linear Equations

```
1  #include <iostream>
2  using namespace std;
3
4  #include "Eigen.hpp"
5
6  int main() {
7    int n=3;
8    using VT=Eigen::vector_t<float>;
9    using MT=Eigen::matrix_t<float>;
10   MT A=MT::Random(n,n);
11   VT b=VT::Random(n);
12   VT x=A.lu().solve(b);
13   cout << x.transpose() << endl;
14   return 0;
15 }
```

yields output

```
0.608759 -0.231281 0.510379
```

- Direct solvers for systems of linear equations

  $$A \cdot x = b,\ A \in \boldsymbol{R}^{n \times n},\ b \in \boldsymbol{R}^n$$

  determine $x \in \boldsymbol{R}^n$ such that $x = A^{-1} \cdot b$, where A^{-1} denotes the inverse of A; conditions apply.
- Different factorizations of A are available including LU, LL^T, and QR factorizations implemented as non-static member functions of `Eigen::matrix_t` (line 12).
- Corresponding solvers are implemented as non-static member functions of the respective factorization (line 12).

Summary

Eigen

Vectors

Matrices

Linear Algebra
    Arithmetic
    Solution of Systems of Linear Equations