TENLAB:
When Matrices Are Not Enough

Mehmet Turkcan, Dallas R. Jones, Cem Subakan

May 11, 2016

TENLAB in One Slide (Or So)

- The name gives away the focus of the language!

TENLAB in One Slide (Or So)

- The name gives away the focus of the language!
- The example:

$$\textit{let } A \in \mathbb{R}^{I_1 \times I_2 \times I_3}, B \in \mathbb{R}^{J_1 \times J_2}, \textit{ in}$$

$$C_{i_1} = \sum_{k_1,k_2} A_{i_1,k_1,k_2} B_{k_1,k_2}$$

TENLAB in One Slide (Or So)

- The name gives away the focus of the language!
- The example:

$$\textit{let } A \in \mathbb{R}^{I_1 \times I_2 \times I_3}, B \in \mathbb{R}^{J_1 \times J_2}, \textit{ in}$$
$$C_{i_1} = \sum_{k_1, k_2} A_{i_1, k_1, k_2} B_{k_1, k_2}$$

- In MATLAB this does not work really well:

```
C = sum(sum(bsxfun(@times,A,shiftdim(B,-1)),3),2);
```

and requires some thinking. Worse if the problem is not trivial.

TENLAB in One Slide (Or So)

- The name gives away the focus of the language!
- The example:

$$\textit{let } A \in \mathbb{R}^{I_1 \times I_2 \times I_3}, B \in \mathbb{R}^{J_1 \times J_2}, \textit{ in}$$

$$C_{i_1} = \sum_{k_1,k_2} A_{i_1,k_1,k_2} B_{k_1,k_2}$$

- In MATLAB this does not work really well:

```
C = sum(sum(bsxfun(@times,A,shiftdim(B,-1)),3),2);
```

  and requires some thinking. Worse if the problem is not trivial.
- What if we could just write:

```
C = {1,1} A {2,3} .* B {1,2};
```

  Now that's some cool imperative laziness!

MATLAB's Solution

bsxfun

R2016a

Apply element-by-element binary operation to two arrays with singleton expansion enabled

collapse all in page

Syntax

```
C = bsxfun(fun,A,B)
```

Description

`C = bsxfun(fun,A,B)` applies the element-by-element binary operation specified by the function handle `fun` to arrays `A` and `B`, with singleton expansion enabled. `fun` can be one of the following built-in functions:

`@plus`	Plus
`@minus`	Minus
`@times`	Array multiply
`@rdivide`	Right array divide
`@ldivide`	Left array divide
`@power`	Array power
`@max`	Binary maximum
`@min`	Binary minimum
`@rem`	Remainder after division
`@mod`	Modulus after division
`@atan2`	Four-quadrant inverse tangent; result in radians
`@atan2d`	Four-quadrant inverse tangent; result in degrees
`@hypot`	Square root of sum of squares
`@eq`	Equal
`@ne`	Not equal
`@lt`	Less than

Summary of TENLAB

- Imperative multi-dimensional array manipulation language.
- Built to address the needs people in Machine Learning or similar disciplines who want to work with multi-dimensional arrays.
- Compiles into C (Fast!).
- Effortlessly interfaces with MATLAB.
- Includes a very powerful Tensor Product implementation.

Generalized Tensor-Tensor Product

$$C_{i_1} = \sum_{k_1,k_2} A_{i_1,k_1,k_2} B_{k_1,k_2}$$

```
C = {1,1} A {2,3} .* B {1,2};
```

Output tensor

Collapse or not collapse

Matched dimensions of A

Matched dimensions of B

TENLAB: Even More Generalized Tensor-Tensor Product

TENLAB is even more flexible:

$$C_{i_1} = \sum_{k_1,k_2} A_{i_1,k_1,k_2} B_{k_1,k_2}$$

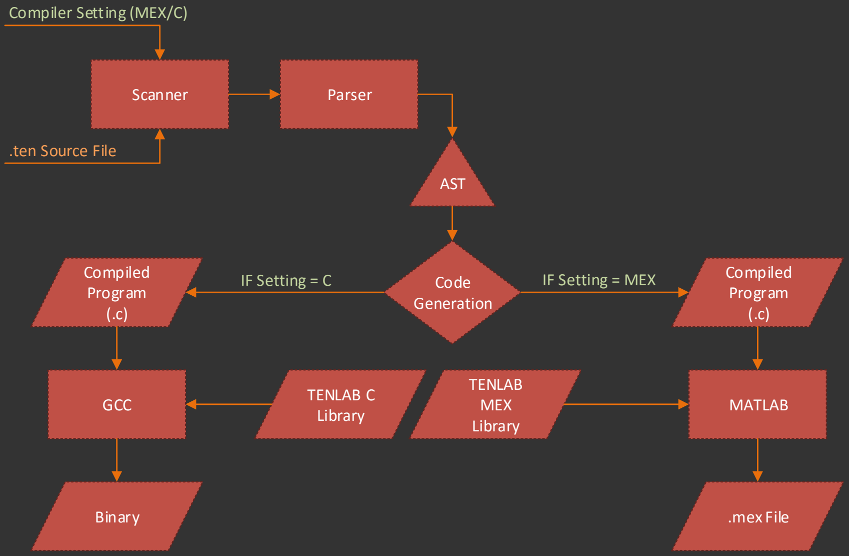
```
C = {1,1} {5,3,7} A {2,3} .* {3,7} B {1,2};
```

- C: Output tensor
- {1,1}: Collapse or not collapse
- {5,3,7}: Shape to use for A
- {2,3}: Matched dimensions of A
- {3,7}: Shape to use for B
- {1,2}: Matched dimensions of B

Current List of Features & Ideology

- Generalized Tensor Product
- Be Like Water (When It Comes to Tensors)
- Memory Safety
- Functions and Scripts
- Full MATLAB Integration (With Multiple Outputs!)

Compiler Architecture



Language Design

Function definitions followed by scripts:

```
% Beginning of Function Declarations
function gcd (Z, X, Y);
  Z = X != Y;
  while (Z);
    if (X > Y);
      X = X - Y;
    else;
      Y = Y - X;
    end;
    Z = X != Y;
  end;
  return X;
end;
% Beginning of the Script
tensor A;
tensor B;
tensor C;
A = 12;
B = 14;
A = gcd(C,A,B);
print(A);
```

Tensor Products

For simplicity, let's consider a matrix product:

```
% Beginning of the Script
tensor X;
tensor Y;
tensor Z;
X = [[3,4],[4,5],[6,7]];
Y = [[1,2,3],[4,5,6]];
Z = [[0,0,0],[0,0,0],[0,0,0]];
Z = {1} {3,2} X {2} .* {2,3} Y {1};
print(Z);
```

MATLAB Integration

MATLAB integration works as follows:

```
% Beginning of the Script
tensor A;
tensor B;
tensor C;
input(A,1);
input(B,2);
input(C,3);
C = {1} {11,8,2} A {3} .* {22,44,2} B {3};
print(C);
output(C,1);
```

Built-in Functions

Design Constraint: Avoid the Standard Library Syndrome, but remain versatile.

- `print` and `shape`: Display Results
- `input` and `output`: Get Data from MATLAB
- `set`, `length`, `pop` and `dequeue`: Change the Content
- `reshape`: Alter the Shape using Content
- `clear` and `clean`: Clear and Clean the Tensors

Testing

/home/tenlab

```
mehme@DESKTOP-ND3IB73 /home/tenlab
$ make
make: 'tenlab' is up to date.

mehme@DESKTOP-ND3IB73 /home/tenlab
$ bash testlab.sh
testlab.sh: line 6: ulimit: cpu time: cannot modify limit: Invalid argument
fail-assign1 Works!
fail-dequeue Works!
fail-for1 Works!
fail-mex1 Works!
fail-pop Works!
fail-print Works!
fail-reshape Works!
fail-while1 Works!
test-arith1  Works!
test-arith2  Works!
test-assign1  Works!
test-assign2  Works!
test-clear1  Works!
test-comment1  Works!
test-comment2  Works!
test-dequeue1  Works!
test-fib  Works!
```

- A total of 61 test cases included.
- 35 for TENLAB, 18 for C libraries, 4 for low-level MEX integration and 4 demos.
- Fundamental to the success of the team, automated test suites were the first to be built.

Lessons Learned

- OCaml is not the enemy.
- Each member of the team should know everything about the codebase.
- Testing and test automation is key. Without automation, we would have had nothing.
- Gained a lot of ideas for future languages and implementation improvements.
- Don't lose hope or panic near the end, keep on going!

Demo

Let's have some bsxfun!

The End: Q&A

Thanks a lot for listening! Any questions?