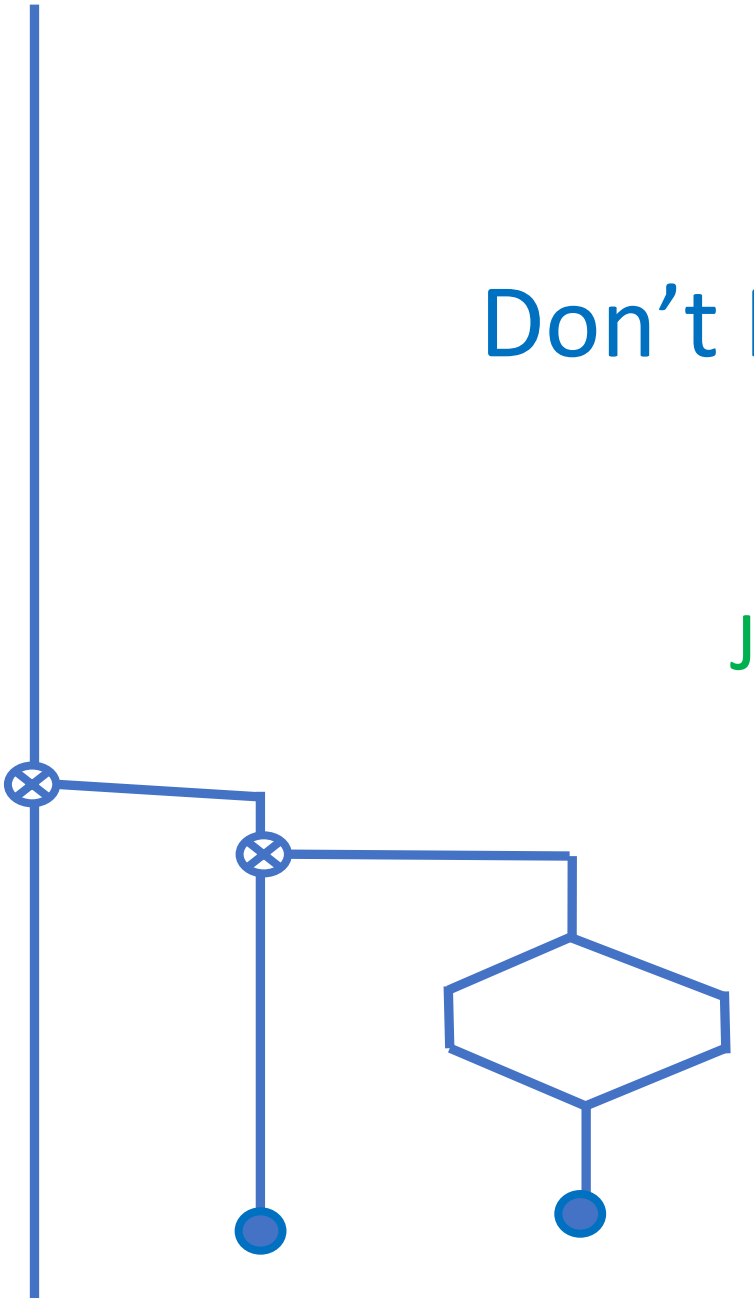
Don’t Fear Deep Stack

SVFIG

June 26, 2021

Bill Ragsdale



Notes

- Matrices require four levels of access:
  - Matrix memory address
  - Cell size, 8 bytes
  - Matrix size in rows and columns.
  - Offsets within the matrix.
- Forth matrix addressing is zero based.

Matrix Realities

- Matrix support requires a large number of parameters.
- They can be hidden at the top levels but expand at the low levels.
- For one matrix 8 parameters are required.
  - Address, cell size, #rows, #columns, rows from:to, columns from:to.
- For arithmetic up to 24 parameters are involved.

Matrix Realities II

- This can be brought down to 5 parameters per matrix at the user level or 15 for arithmetic.
- You can use 15 variables.
- Or else 15 locals.
- Or else do deep stack manipulation within a small number of low level words.
- We will examine these levels.

The Starting Point

```
3 3 create{ a{
          a{ }fill
          a{ }list
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000
```

Our Initial Values

```
a{ }list
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000
```

```
b{ }list
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000
```

```
c{ }list
```

```
.00000 .00000 .00000
.00000 .00000 .00000
.00000 .00000 .00000
```

Sum Full Matrices, }+

```
a{ b{ c{ }+

: }+
    3}expand  }sub+  ;
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 2.0000 4.0000
20.000 22.000 24.000
40.000 42.000 44.000
```

Note: 3 input parameters are expanded to 15.

Sum A Common Row, }r+

```
a{ b{ c{ 1 }r+
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000
```

```
: }r+
 >r rot r@ dup >cols<
 6 roll r@ dup >cols<
10 roll r> dup >cols<
   }sub+ ;
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 .00000 .00000
20.000 22.000 24.000
.00000 .00000 .00000
```

Note: 4 input parameters are expanded to 15.

Sum Independent Rows, }rrr+

```
  a{ 0  b{ 1  c{ 2 }rrr+
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000
```

```
: }rrr+
 2>r    2swap  dup >cols<
 6 roll 6 roll dup >cols<
 2r>           dup >cols<
  }sub+ ;
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 .00000 .00000
.00000 .00000 .00000
10.000 12.000 14.000
```

Note: 6 input parameters are expanded to 15.

Sum A Common Column, }c+

```
a{ b{ c{ 1 }c+

: }c+
 >r rot  >rows< r@ dup
  6 roll >rows< r@ dup
 10 roll >rows< r> dup
 }sub+ ;
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 2.0000 .00000
0.0000 22.000 0.0000
.00000 42.000 .00000
```

Note: 4 input parameters are expanded to 15.

Sum Independent Columns, }ccc+

```
  a{ 0  b{ 1  c{ 2 }rrr+
```

```
: }ccc+
 2>r  2swap  >r >rows<
 r> dup 6 roll 6 roll
              >r >rows<
 r> dup 2r>  >r >rows<
 r> dup  }sub+ ;
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 .00000 1.0000
.00000 .00000 21.000
.00000 .00000 41.000
```

Note: 6 input parameters are expanded to 15.

Sum A Single Cell

```
a{ b{ c{ 1 1 }rc+


: }rc+
  swap dup rot dup
  }com+ ;


Note: 7 input parameters are
expanded to 15.
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 .00000 .00000
.00000 22.000 .00000
.00000 .00000 .00000
```

Sum Common Sub-Matrices

```
a{ b{ c{ 1 2 1 2 }com+


: }com+
 2over 2over 2>r 2>r
 2over 2over 2>r 2>r
 5 roll      2r> 2r>
 9 roll      2r> 2r>
  }sub+  ;
```

Note: 7 input parameters are expanded to 15.

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 .00000 .00000
.00000 22.000 24.000
.00000 42,000 44.000
```

Sum Independent Sub-Matrices

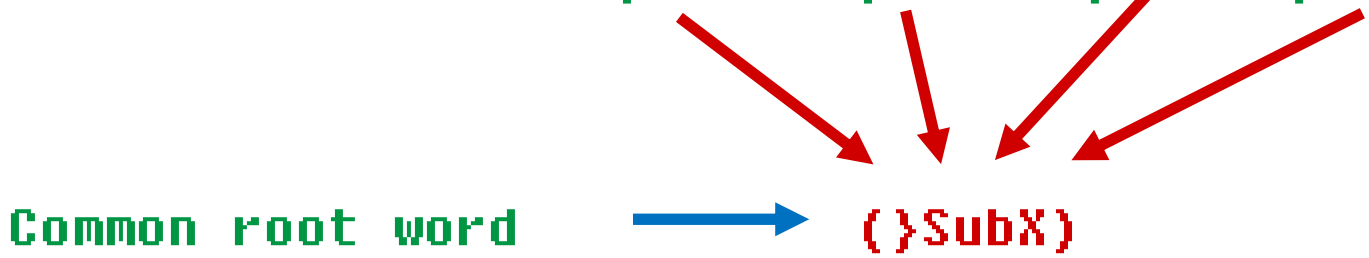
```
a{ 0 1 0 1
b{ 0 1 1 2
c{ 1 2 1 2   }sub+
```

```
.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 1.0000 2.0000
10.000 11.000 12.000
20.000 21.000 22.000

.00000 .00000 .00000
.00000 1.0000 3.0000
.00000 21.000 23.000
```

Here we input all 15 parameters for addition.

All Math Operators

Matrix	}+	}-	}.*	}./
Common row	}r+	}r-	}r.*	}r./
Independent rows	}rrr+	}rrr-	}rrr.*	}rrr./
Common column	}c+	}c-	}c.*	}c./
Independent columns	}ccc+	}ccc-	}ccc.*	}ccc./
Single cell	}rc+	}rc-	}rc.*	}rc./
Common sub-cell	}com+	}com-	}com.*	}com./
Independent sub-cells	}sub+	}sub-	}sub.*	}sub./

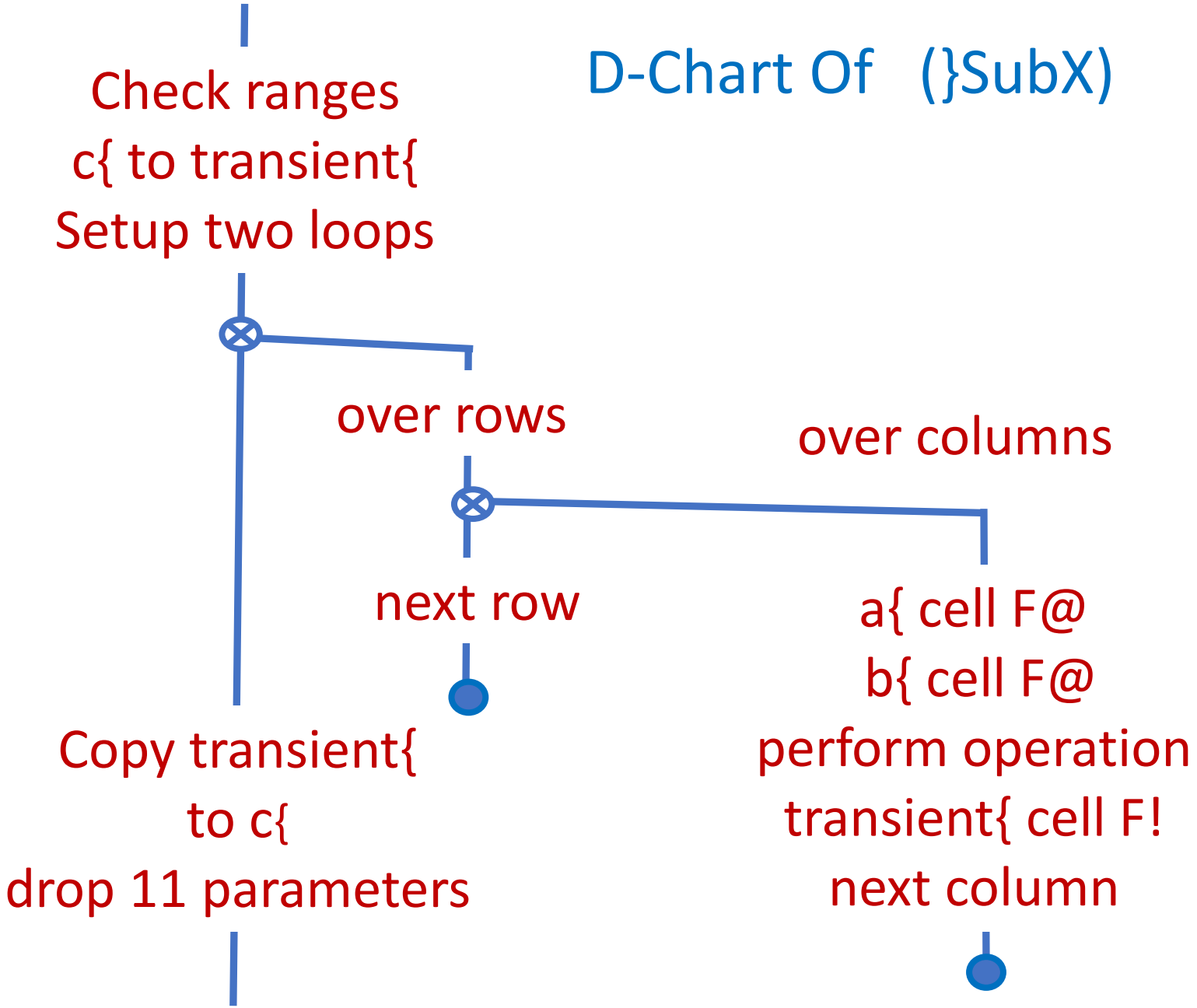


Note: }.*, }./ (et. al.) are elementwise, not matrixwise.

(}SubX) Pseudocode

```
(}SubX)

Check input ranges
Copy c{ to transient{
Over all rows and columns
     retrieve cell in a{
     retrieve cell in b{
     perform math operation
     store in cell in transient{
   loop over all columns
 loop over all rows
Copy transient{ to c{
Clean up data stack
```

D-Chart Of (}SubX)



Master Math Operator (}SubX)

```
a{ r1 r2 c1 c2 b{ r3 r4 c3 c4 c{ r5 r6 c5 c6  ['] F+ (}SubX)


: (}SubX)
  14 pick 14 pick 11 pick 11 pick 8 pick 8 pick 3RangeValid?
  12 pick 12 pick  9 pick  9 pick 6 pick 6 pick 3RangeValid?
   5 pick dup }dimensions opentransient{ transient{ }RawCopy
  13 roll 12 roll 2drop  8 roll 7 roll 2drop
    swap  2 pick - 1+    3 roll 4 pick - 1+
    0 do dup 0 do 10 pick {{ 13 pick j +  13 pick i + }}@
                  7 pick {{ 10 pick j +  10 pick i + }}@
  over execute transient{ {{  7 pick j +   7 pick i + }}!
  loop loop
  transient{ 5 pick  }copy  closetransient{
  4drop 4drop 3drop  ;

   Execution: adding 1,000 matrices = 16 milliseconds.
```

Comments

Routine manipulations and math are straight forward and brief.

Only one word needs all 15 input parameters.

The alternative is to unload those parameters into variables or values and then reload when needed.

The major effort creating those few, complex words saves the specific coding and space for many words.

Benefits

In MatLab, Octave, etc. the elaborate syntax could be, hypothetically:

```
C[3:end, 1:2]  = A[var:end, 4:5](3:6)
               + B[1:5, j:k](j k l m).
```

Matrix Forth replaces this syntax with many single function words each with a crystal clear syntax. Plus, you may add your own words.

And, testing becomes extremely easy.

Credits

- Dr. Julian Noble for the key concepts of matrix structure and its parameter use. See *Scientific FORTH*
- Andrew McKewan and Tom Zimmer for Win32Forth.
- The European team who updated it in the early 2000s.

References

- https://github.com/BillRagsdale/Matrix-Forth-Wordset
- https://github.com/BillRagsdale/WIN32Forth-Guide