m2cpp – A MATLAB to C++ Code Translator with Traceability

Geir Y. Paulsen*, Bjørn Nordmoen, Xing Cai, Stuart Clark, Jonathan Feinberg and Hans Petter Dahle
**Corresponding author*, email: geirpa@simula.no

Seismic Surveying: The Need for Optimised Code and Agile Development

Optimisation

- Huge Data Rates (> 1 Gb/s)
- Computing Power (> 2 Tflop)
- Network of Computers (>2000 cores)
- But it is difficult to write optimised code quickly for new algorithms

Agility

- MATLAB is familiar to seismologists
- It is easy to write new algorithms
- However, runtime is too slow for industrial execution!

The Idea: Legible Code Translation

MATLAB

```
B = M'*M;
beta = B(1,1)*mu/100;
ab = (B + beta*eye(lf))\M'*y;
temp = M*ab;
temp = [temp;zeros(lf,1)];
```

m2cpp

C++/Armadillo

```
B = arma::trans(M)*M ;
beta = B(0, 0)*mu/100.0;
ab =
arma::solve((B+beta*arma::eye<cx_mat>(If,If)
), arma::trans(M), solve_opts::fast)*y;
temp = M*ab;
temp = arma::join_cols(temp,
arma::zeros<cx_mat>(If, 1));
```

MATLAB

```
figure(1);
imagesc([1:1:3*nx],[0:1:nt-
1]*dtsec,clip([D,D1,D-D1],50,50));
colormap(seismic(1));
```

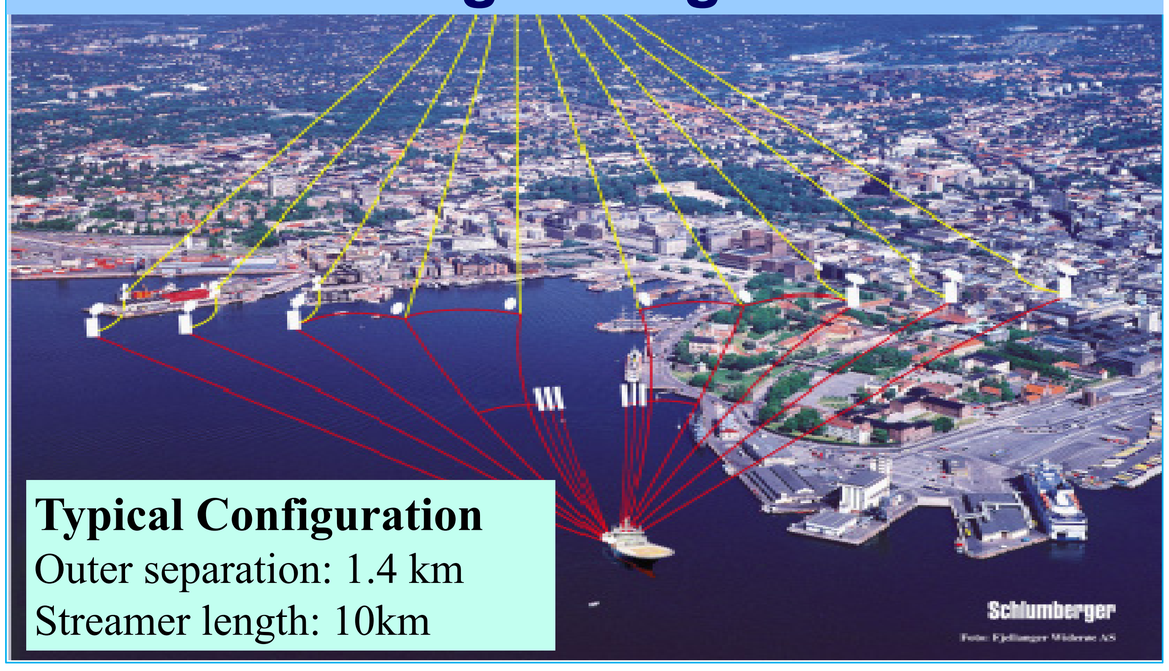
m2cpp

C++/Armadillo

```
_plot.figure(1);
_plot.imagesc(
m2cpp::span<rowvec>(1, 1, 3*nx),
m2cpp::span<rowvec>(0, 1, nt-1)*dtsec,
clip({arma::join_rows(arma::join_rows(D,
D1), D-D1)}, 50, 50)
);
_plot.colormap(seismic(1));
```

Try it! m2cpp is on github:
https://github.com/emc2norway/m2cpp

Seismic Towing Configuration



Typical Configuration
Outer separation: 1.4 km
Streamer length: 10km

Workflow

MATLAB Source
- Parse the code and create an Abstract Syntax Tree

C++ with Armadillo
- Armadillo allows C++ Syntax to be similar to MATLAB

Parallelisation and Vectorisation
- Currently supports parallelising loops using OpenMP or TBB

[Optional] Hand-coded optimisation
- Since the code is readable, manual optimisation is possible

MATLAB to C++ Speed up (Single Core)

We tested the execution times and speed-up factor of the generated C++ from m2cpp vs the MATLAB on 5 examples (single core).

Code Example	Problem Size	MATLAB Time (s)	C++ Time (s)	Speed-up
fx_decon	3.2 x 10^6	2.7	1.9	**1.4**
moveout	1.0 x 10^7	3.0	0.9	**3.3**
parabolic	6.3 x 10^6	268	141	**1.9**
radon decon	6.3 x 10^6	34	27	**1.3**
va	2.5 x 10^6	172	70	**2.5**

Targeting Parallelism: TBB for Loops

```
%#TBB
for j = 1:num_points
   y1(j) = sin(x(j));
end
```

m2cpp

%#TBB tells m2cpp to parallelise the for loop with TBB

```
tbb::parallel_for(
     tbb::blocked_range<size_t>(1, num_points+1),[&x, &y1](const
     tbb::blocked_range<size_t>& _range)
    {
     for (uword j = _range.begin(); j != _range.end(); j++)
     {
       y1(j-1) = sin(x(j-1)) ;
     }
    }
);
```

Conclusion: m2cpp is a traceable translator that creates faster code

- Code in C++ appears similar to MATLAB code
- We obtain speed-up factors from 1.4 to over 3 on a single core
- Hand-coded optimisations are easier to implement because generated C++ code is readable
- Multi-core parallelisation available via OpenMP or TBB

Contributing Partners:
WesternGeco Schlumberger simula FORNEBU CONSULTING A PART OF DEVOTEAM ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership