MEX Compilation

Table of Contents

Introduction

The MATLAB scripts in "./IVA/algo" are designed to support compilation into MEX files.

When the number of channels is two, compiling to MEX generally improves execution speed across all algorithms. For cases with more channels, MEX compilation is essential for `algo_IVA_AuxISS.m` (which contains conditional branches), as it brings its performance closer to `algo_IVA_AuxISS_without_if_statements.m`.

This document outlines the procedure for building the MEX files and provides example code for runtime benchmarking.

Building MEX file

Add path to "./IVA" and prepare the mixture.

```
addpath(genpath("./IVA"))
[signal1, signal2, fs] = util_loadSampleMixture;
mixture                = signal1 + signal2;
```

Check your compiler. We recommend `Microsoft Visual C++ 2022 (C)`.

```
mex -setup
```

Call `buildmex_.*` for easy compilation.

These functions generate C code tailored to the types of the input variables and build the corresponding MEX file.

Note that a rebuild is required if the array size (e.g., the number of channels or the length of `mixture`) changes. To support variable-length signals or change compilation options, you can either modify the `buildmex_.*` functions or compile manually (see next section).

```
buildmex_IVA_FastADMM(mixture);
buildmex_IVA_ADMM(mixture);
buildmex_IVA_PDS(mixture);
buildmex_IVA_AuxIP(mixture);
buildmex_IVA_AuxISS(mixture);
buildmex_IVA_AuxIP2(mixture)
```

Quickly compare the runtime before and after MEX compilation.

Note that the measurements include preprocessing and postprocessing steps such as STFT, iSTFT, and algorithm-specific matrix reshaping. For a more accurate and fair comparison, see the next section.

```
addpath(genpath("./IVA"))
```

FastADMM-IVA

```
tic; run_IVA_FastADMM(mixture); toc;
tic; runmex_IVA_FastADMM(mixture); toc;
```

ADMM-IVA

```
tic; run_IVA_ADMM(mixture); toc;
tic; runmex_IVA_ADMM(mixture); toc;
```

PDS-IVA

```
tic; run_IVA_PDS(mixture); toc;
tic; runmex_IVA_PDS(mixture); toc;
```

AuxIVA-IP

```
tic; run_IVA_AuxIP(mixture); toc;
tic; runmex_IVA_AuxIP(mixture); toc;
```

AuxIVA-ISS

```
tic; run_IVA_AuxISS(mixture); toc;
tic; runmex_IVA_AuxISS(mixture); toc;
```

AuxIVA-IP2

```
tic; run_IVA_AuxIP2(mixture); toc;
tic; runmex_IVA_AuxIP2(mixture); toc;
```

Runtime Benchmark

For a fair runtime benchmark, directly call the `algo_.*` functions and their MEX-compiled counterparts.

```
addpath(genpath("./IVA"))

T          = 169;
F          = 1025;
numTrial   = 1; % 10
NList      = 2:16;
numMethods = 7;
labels     = ["FastADMM","ADMM","PDS","IP","ISS","ISS w/o if","IP2"];

median_runtime_MATLAB = nan(numMethods, numel(NList));
```

```
median_runtime_MEX    = nan(numMethods, numel(NList));
figure("Visible","on","WindowState","maximized")

for idxN = 1:numel(NList)
    N  = NList(idxN);
    X  = complex(randn(N,T,F), randn(N,T,F));
    Xp = permute(X,[3 2 1]);

    % build MEX files
    codegen algo_IVA_FastADMM                     -silent -d codegen -args
{coder.typeof(X),  200, 1.1, 1.1, 1.1}      -O disable:inline -o
algomex_benchmark_IVA_FastADMM
    codegen algo_IVA_ADMM                         -silent -d codegen -args
{coder.typeof(X),  200, 1.1, 1.1, 1.1}      -O disable:inline -o
algomex_benchmark_IVA_ADMM
    codegen algo_IVA_PDS                          -silent -d codegen -args
{coder.typeof(X),  200, 1.1, 1.1, 1.1, 1.1} -O disable:inline -o
algomex_benchmark_IVA_PDS
    codegen algo_IVA_AuxIP                        -silent -d codegen
-args {coder.typeof(X ), 200}                     -O disable:inline -o
algomex_benchmark_IVA_AuxIP
    codegen algo_IVA_AuxISS                       -silent -d codegen
-args {coder.typeof(Xp), 200}                     -O disable:inline -o
algomex_benchmark_IVA_AuxISS
    codegen algo_IVA_AuxISS_without_if_statements -silent -d codegen
-args {coder.typeof(Xp), 200}                     -O disable:inline -o
algomex_benchmark_IVA_AuxISS_without_if_statements
    codegen algo_IVA_AuxIP2                       -silent -d codegen
-args {coder.typeof(X ), 100}                     -O disable:inline -o
algomex_benchmark_IVA_AuxIP2

    runtime_MATLAB_n = nan(numTrial, numMethods);
    runtime_MEX_n    = nan(numTrial, numMethods);

    for trial = 1:numTrial
        % measure runtime
        tic; algo_IVA_FastADMM                    (X, 200,1,1,1);
runtime_MATLAB_n(trial,1) = toc;
        tic; algo_IVA_ADMM                        (X, 200,1,1,1);
runtime_MATLAB_n(trial,2) = toc;
        tic; algo_IVA_PDS                         (X, 200,1,1,1,1);
runtime_MATLAB_n(trial,3) = toc;
        tic; algo_IVA_AuxIP                       (X, 200);
runtime_MATLAB_n(trial,4) = toc;
        tic; algo_IVA_AuxISS                      (Xp,200);
runtime_MATLAB_n(trial,5) = toc;
        tic; algo_IVA_AuxISS_without_if_statements(Xp,200);
runtime_MATLAB_n(trial,6) = toc;
        tic; algo_IVA_AuxIP2                      (X, 100);
runtime_MATLAB_n(trial,7) = toc;
```

```
        tic; algomex_benchmark_IVA_FastADMM                    (X, 200,1,1,1);
runtime_MEX_n(trial,1) = toc;
        tic; algomex_benchmark_IVA_ADMM                        (X, 200,1,1,1);
runtime_MEX_n(trial,2) = toc;
        tic; algomex_benchmark_IVA_PDS                         (X, 200,1,1,1,1);
runtime_MEX_n(trial,3) = toc;
        tic; algomex_benchmark_IVA_AuxIP                       (X, 200);
runtime_MEX_n(trial,4) = toc;
        tic; algomex_benchmark_IVA_AuxISS                      (Xp,200);
runtime_MEX_n(trial,5) = toc;
        tic; algomex_benchmark_IVA_AuxISS_without_if_statements(Xp,200);
runtime_MEX_n(trial,6) = toc;
        tic; algomex_benchmark_IVA_AuxIP2                      (X, 100);
runtime_MEX_n(trial,7) = toc;

        % visualize
        median_runtime_MATLAB(:,idxN) = median(runtime_MATLAB_n,1,"omitmissing");
        median_runtime_MEX   (:,idxN) = median(runtime_MEX_n   ,1,"omitmissing");

        cla;
        plot(NList(1:idxN), median_runtime_MATLAB(:,1:idxN).',"o-","LineWidth",2);
hold on;
        plot(NList(1:idxN), median_runtime_MEX   (:,1:idxN).',"x:","LineWidth",2);

        xscale("log"); yscale("log")
        xlabel("Number of sources"); xticks(2:16);
        ylabel("Runtime [sec]")
        legend(labels(:)+[" (MATLAB)" " (MEX)"], "Location","southoutside",
"Orientation","horizontal", "NumColumns", numMethods);
        drawnow;
    end
end
```

Reference

1. Hiroko Watarai, Kazuki Matsumoto, Kohei Yatabe, "Fast and flexible algorithm for determined blind source separation based on alternating direction method of multipliers" (2025).