Software Outlook: FFT Benchmarks for Fortran Codes

H. Sue Thorne

November 30, 2018

1 Introduction

As part of the 2018/19 Software Outlook Work Plan, we will be benchmarking a number of different Fast Four Transform (FFT) libraries with bindings for Fortran. The attributes of the different libraries are given in Section 5.

2 1D Benchmark

Let A be a 2D array with dimensions $n_1 \times n_2$ and there be q 2D arrays B_i that are the same size as A and satisfy $\sum_i^q [B_i(j,k)]^2 = 1$ for each $j = 1, \dots, n_1$ and $k = 1, \dots, n_2$. The general benchmark will take the form of Algorithm 1, where *comp_mult*(A, B_i) is defined to be component-wise multiplication of A with B_i; *comp_div*(H_i, B_i) is defined to be component-wise division of H_i by B_i; `FFT` is the discrete Fast Fourier Transform and `IFFT` is the discrete inverse Fast Fourier Transform.

3 2D Benchmark

Let A be a 3D array with dimensions $n_1 \times n_2 \times n_3$ and there be q 3D arrays B_i that are the same size as A and satisfy $\sum_i^q [B_i(j,k,l)]^2 = 1$ for each $j = 1, \dots, n_1$, $k = 1, \dots, n_2$ and $l = 1, \dots, n_3$. The benchmark will take the form of Algorithm 3, where *comp_mult*(A, B_i) is defined to be component-wise multiplication of A with B_i; *comp_div*(H_i, B_i) is defined to be component-wise division of H_i by B_i; `FFT` is the discrete Fast Fourier Transform and `IFFT` is the discrete inverse Fast Fourier Transform. This benchmark is designed to imitate some of the workload done in CCP_PETMR's SIRF code.

4 3D Benchmark

Let A be a 3D array with dimensions $n_1 \times n_2 \times n_3$ and there be q 3D arrays B_i that are the same size as A and satisfy $\sum_i^q [B_i(j,k,l)]^2 = 1$ for each $j = 1, \dots, n_1$, $k =$

Algorithm 1 1D Benchmark

for $i = 1, \ldots, q$ **do**
  $C_i = comp\_mult(A, B_i)$
  for $k = 1, \ldots, n_2$ **do**
    $D_k = C_i(:, k)$
    $F_k = \texttt{FFT}(D_k)$
    if do_inverse **then**
      $G_k = \texttt{IFFT}(F_k)$
      $H(:, k) = G_k$
    end if
  end for
  if do_inverse **then**
    $J_i = comp\_div(H_i, B_i)$
    $abs\_err = \|A - J_i\|_2$
  end if
end for

Algorithm 2 2D Benchmark

for $i = 1, \ldots, q$ **do**
  $C_i = comp\_mult(A, B_i)$
  for $l = 1, \ldots, n_3$ **do**
    $D_l = C_i(:, :, l)$
    $F_l = \texttt{FFT}(D_l)$
    if do_inverse **then**
      $G_l = \texttt{IFFT}(F_l)$
      $H(:, l) = G_l$
    end if
  end for
  if do_inverse **then**
    $J_i = comp\_div(H_i, B_i)$
    $abs\_err = \|A - J_i\|_2$
  end if
end for

$1, \ldots, n_2$ and $l = 1, \ldots, n_3$. The benchmark will take the form of Algorithm 3, where $comp\_mult(A, B_i)$ is defined to be component-wise multiplication of A with B_i; $comp\_div(H_i, B_i)$ is defined to be component-wise division of H_i by B_i; `FFT` is the discrete Fast Fourier Transform and `IFFT` is the discrete inverse Fast Fourier Transform.

Algorithm 3 2D Benchmark

for $i = 1, \ldots, q$ **do**
    $C_i = comp\_mult(A, B_i)$
    $F_i = \texttt{FFT}(C_i)$
    if do_inverse **then**
        $H_i = \texttt{IFFT}(F_i)$
        $J_i = comp\_div(H_i, B_i)$
        $abs\_err = \|A - J_i\|_2$
    end if
end for

5 FFT Libraries