ECE 595, Section 10
Numerical Simulations
Lecture 13: Programming with FFTW

Prof. Peter Bermel
February 8, 2013
Outline

• Recap from Wednesday
• Rationale for FFTW
• Planning DFTs
• Executing DFTs
  – Basic interface
  – Advanced interface
• Application examples
Recap from Wednesday

• Real FFTs
• Multidimensional FFTs
• Applications:
  – Correlation measurements
  – Filter diagonalization method
Rationale for FFTW

• In past, most codes focused exclusively on data sets of length $2^m$
• Required padding can $\rightarrow$ 2x runtime
• Processing pure real data can $\rightarrow$ 2x runtime
• Ignoring symmetry/anti-symmetry $\rightarrow$ 2x runtime
• How do we account for all of these possibilities with a single software package?
Planning in FFTW

• “Most people don’t plan to fail; they fail to plan” – John L. Beckley

• Planning our FFT’s before we perform them can make an enormous difference

• FFTW uses a set of short codes, or “codelets,” which can be called as needed by the planner

• FFTW also compares the different possibilities using dynamic programming
Planning in FFTW

• Execution time can be found in different ways:
  – Estimate: uses heuristics to roughly determine
  – Measure: makes direct test runs with multiple candidate plans

• Execution time may not be directly related to the number of operations

• Instruction-level parallelism can play a critical role in enhancing performance – for example: SIMD
Planning in FFTW

```c
#include <fftw3.h>
...
{
    fftw_complex *in, *out;
    fftw_plan p;
    ...
    in = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
    out = (fftw_complex*) fftw_malloc(sizeof(fftw_complex) * N);
    p = fftw_plan_dft_1d(N, in, out, FFTW_FORWARD, FFTW_ESTIMATE);
    fftw_execute(p);
}```

Sign in exponent

Method of estimating execution time
1D Real DFT's

```c
#include <fftw3.h>
...
{
    double *in, *final;
    fftw_complex *out;
    fftw_plan p1, p2;
    ...
    p1 = fftw_plan_dft_r2c_1d(N, in, out, FFTW_MEASURE);
    p2 = fftw_plan_dft_c2r_1d(N, out, final, FFTW_MEASURE);
    fftw_execute(p1);
    fftw_execute(p2);
```

Multidimensional Real DFTs

```c
#include <fftw3.h>
...
{
    double *in, *final;
    fftw_complex *out;
    fftw_plan p1, p2;
    ...
    p1 = fftw_plan_dft_r2c_2d(n0, n1, in, out, FFTW_PATIENT);
    p2 = fftw_plan_dft_c2r_2d(n0, n1, out, final, FFTW_PATIENT);
    fftw_execute(p1);
    fftw_execute(p2);
```
Multidimensional Complex DFT’s

```c
#include <fftw3.h>
...
{
    double fftw_complex *in, *out, *final;
    fftw_plan p1, p2;
    ...
    p1 = fftw_plan_dft_2d(n0, n1, in, out, FFTW_EXHAUSTIVE);
    p2 = fftw_plan_dft_2d(n0, n1, out, final, FFTW_EXHAUSTIVE);
    fftw_execute(p1);
    fftw_execute(p2);
...```

Method of estimating execution time

2D forward transform

2D backwards transform (un-normalized)
Learning from Your Experience

• **Wisdom** allows one to compute good plans once and save them to disk:
  ```c
  fftw_export_wisdom_to_filename("wise-dft.wis");
  ```

• Can then restore the wisdom next time with:
  ```c
  fftw_import_wisdom_from_filename("wise-dft.wis");
  ```

• While wisdom accumulates over time, one can discard it with:
  ```c
  fftw_forget_wisdom();
  ```
Example: Beam Propagation

• Starting from the Helmholtz equation:
  \[-\nabla^2 \psi = \left( \frac{n\omega}{c} \right)^2 \psi\]

• One can assume a solution of the form:
  \[\psi = \phi e^{-j\beta z}\]

• Where \(\phi\) is slowly varying, which gives rise to:
  \[-\nabla^2 \phi + 2j\beta \hat{z} \cdot \nabla \phi = k_\perp^2 \phi\]
Example: Beam Propagation

• BPM closely resembles the nonlinear Schrodinger equation, which describes a broad class of problems
• For now, we’ll focus on direct applications in optics
• Can solve in real-space or Fourier-space
Example: Beam Propagation

\[
[xx, yy] = \text{meshgrid}([xa:del:xb-del], [1:1:zmax]);
\]

\[
\text{mode} = A*\exp(-((x+x0)/WO).^2); \quad \% \text{Gaussian pulse}
\]

\[
dftmode = \text{fix}(\text{fft}(\text{mode})); \quad \% \text{DFT of Gaussian pulse}
\]

\[
zz = \text{imread('ybranch.bmp','BMP'); \% Upload image with the profile}
\]

\[
\text{phase1} = \exp((i*deltaz*kx.^2)./(nbar*k0 + \sqrt{\text{max}(0,nbar^2*k0*2 - kx.^2)}));
\]

\[
\text{for } k = 1:zmax,
\]

\[
\text{phase2} = \exp(-(od + i*(n(k,:) - nbar)*k0)*deltaz);
\]

\[
\text{mode} = \text{ifft}((\text{fft}(\text{mode}).*\text{phase1})).*\text{phase2};
\]

\[
zz(k,:) = \text{abs}(%mode);
\]

end
Example: Beam Propagation
Next Class

- Is on Monday, Feb. 11
- Will discuss beam propagation method
- Recommended reading: Obayya, Sections 2.2-2.6