Transformacja Fouriera i biblioteka CUFFT 3.0

Procesory Graficzne w Zastosowaniach Obliczeniowych

Karol Opara

Warszawa, 14 kwietnia 2010

Transformacja Fouriera

Definicje i Intuicje

- Transformacja z dziedziny czasu w dziedzinę częstotliwośći $F : R^R \rightarrow C^R$
- Oznaczenia: $F[f(x)](\xi)$, $\hat{f}(\xi)$
- Rozkład funkcji (sygnału) na szereg funkcji okresowych

$$F[f(x)](\xi) = \hat{f}(\xi) := \int_{-\infty}^{\infty} f(x)\ e^{-2\pi i x \xi}\, dx$$

- $cos(2\pi\xi) + i\, sin(2\pi\xi) = e^{2\pi i \xi}$
- Istnieje transformata odwrotna

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)\ e^{2\pi i x \xi}\, d\xi$$

- Jeśli x reprezentuje czas [s], to ξ reprezentuje częstotliwość [Hz]

Dyskretna Transformacja Fouriera (DFT)

- Sygnał próbkowany (np. dzwięk, sygnały elektromagnetyczne, obrazki) opisany próbkami $(a_0, a_1, ..., a_{n-1})$, gdzie $a_i \in R$
- Ciąg harmonicznych $(A_0, A_1, ..., A_{n-1})$, gdzie $A_i \in C$

$$A_k = \sum_{n=0}^{N-1} a_n exp\left(\frac{2\pi i}{N}\right)^{-kn}$$

- Istnieje przekształcenie odwrotne

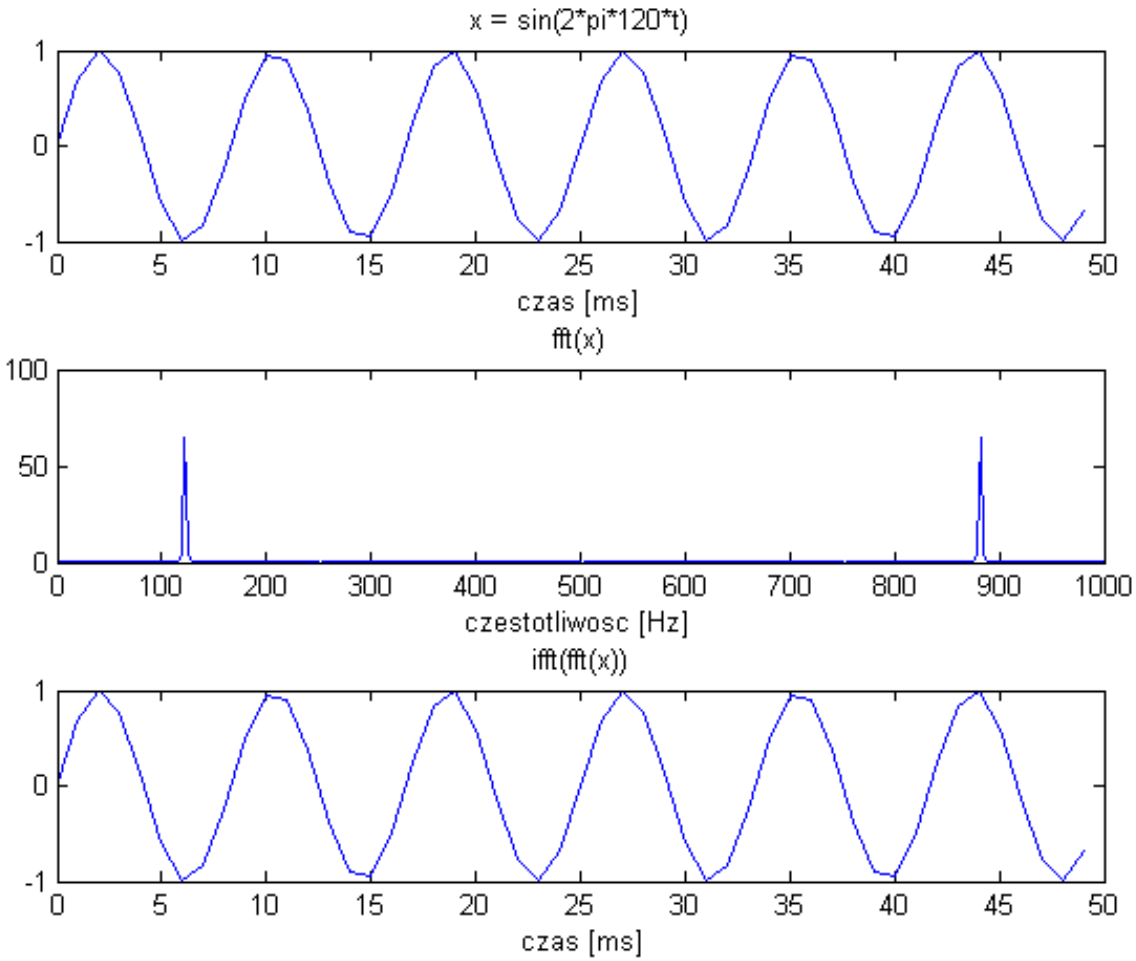
$$a_n = \frac{1}{N} \sum_{k=0}^{N-1} A_k exp\left(\frac{2\pi i}{N}\right)^{kn}$$

- Złożoność algorytmu naiwnego $O(n^2)$

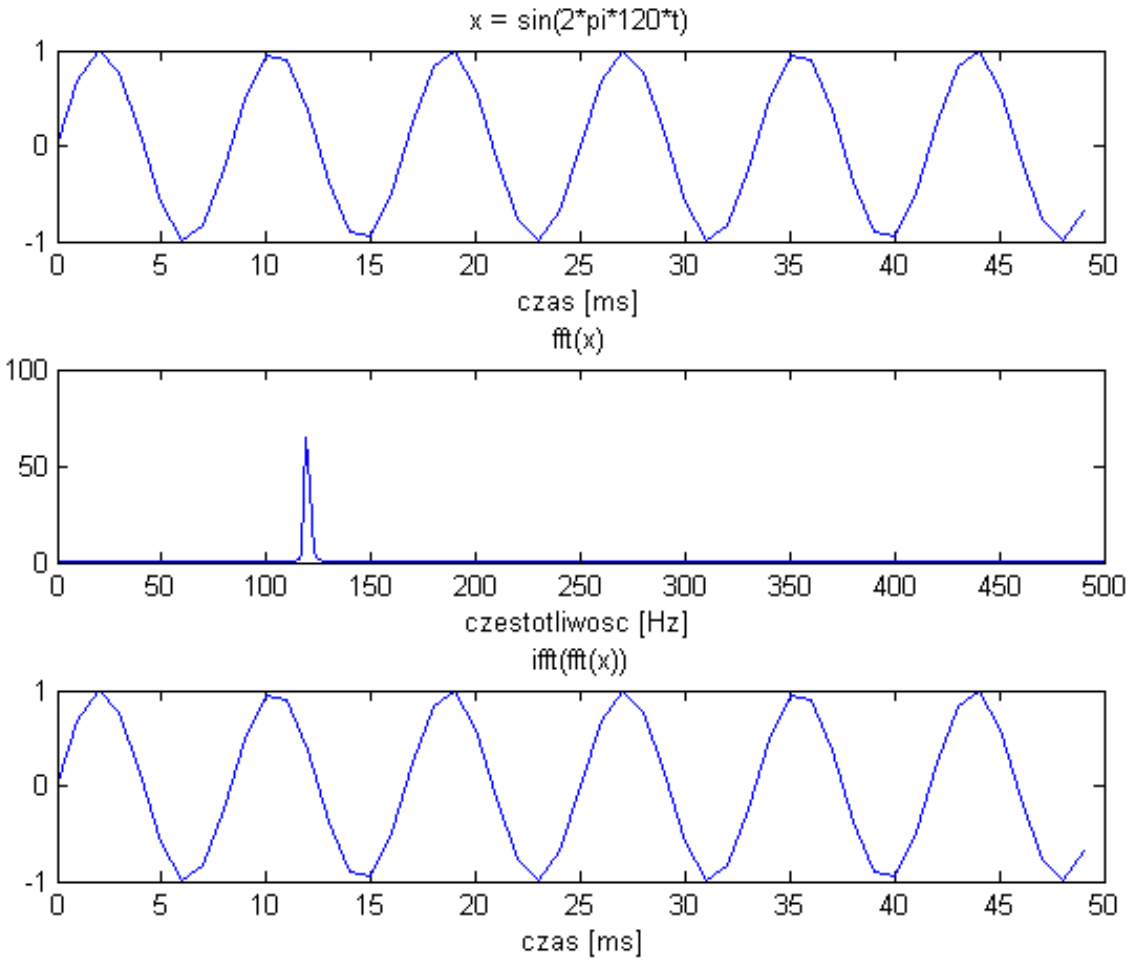
Przykłady transformat sygnałów

- Sygnał próbkowany z częstotliwością 1000 Hz w czasie 0.5 s
- Dyskretna transformata Fouriera (DFT) oparta na 512 punktach (ostatnie 12 punktów zapełnione zerami)
- Do poprawnego odtworzenia sygnału trzeba próbkować przynajmniej dwa razy na okres (tw. Kotielnikowa-Shannona daje ograniczenie na częstotliwość poprawnie odtwarzanych sygnałów)
- Moc sygnału $x(t)$ dla częstotliwości ν dana jest przez $\hat{x}(\nu)\overline{\hat{x}(\nu)}$, czyli kwadrat modułu transformaty

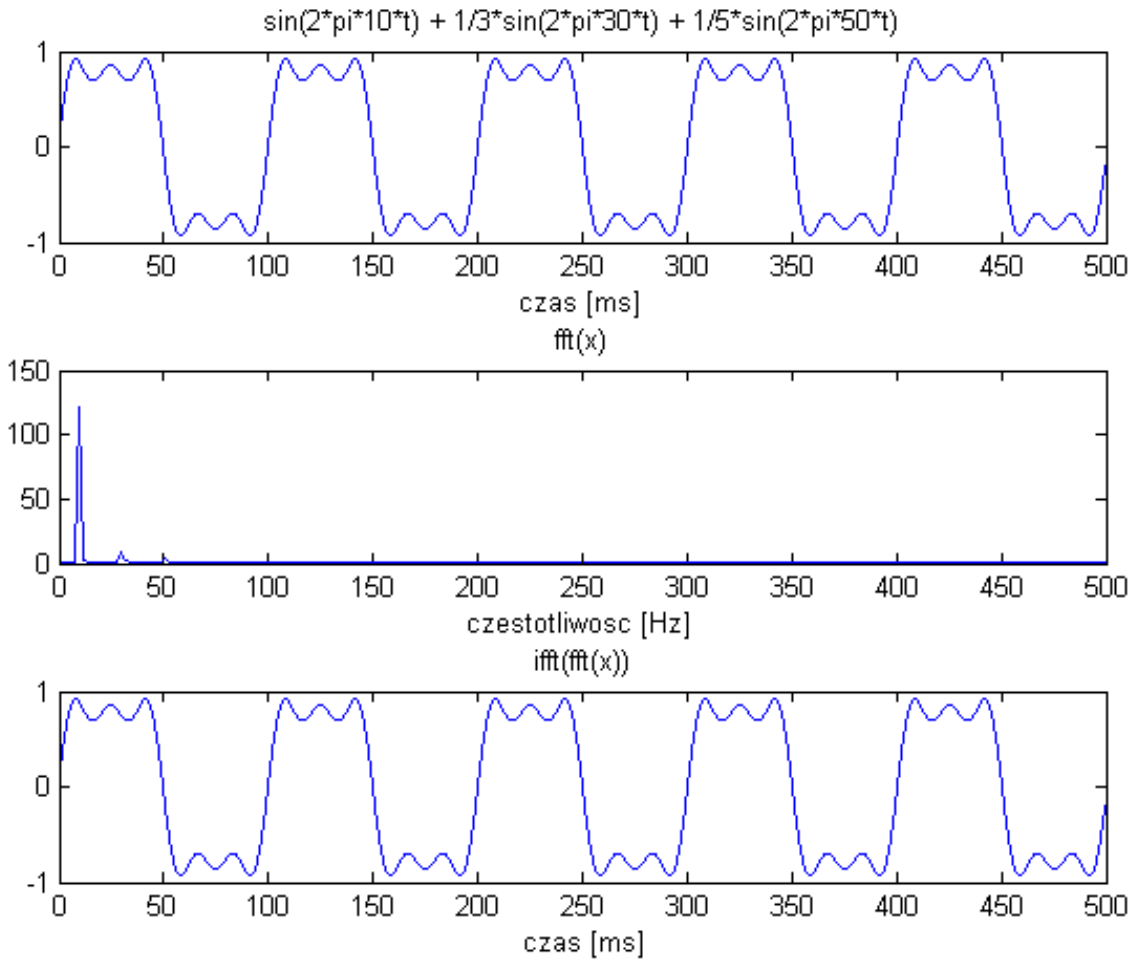
Przykład: transformacja sinusa



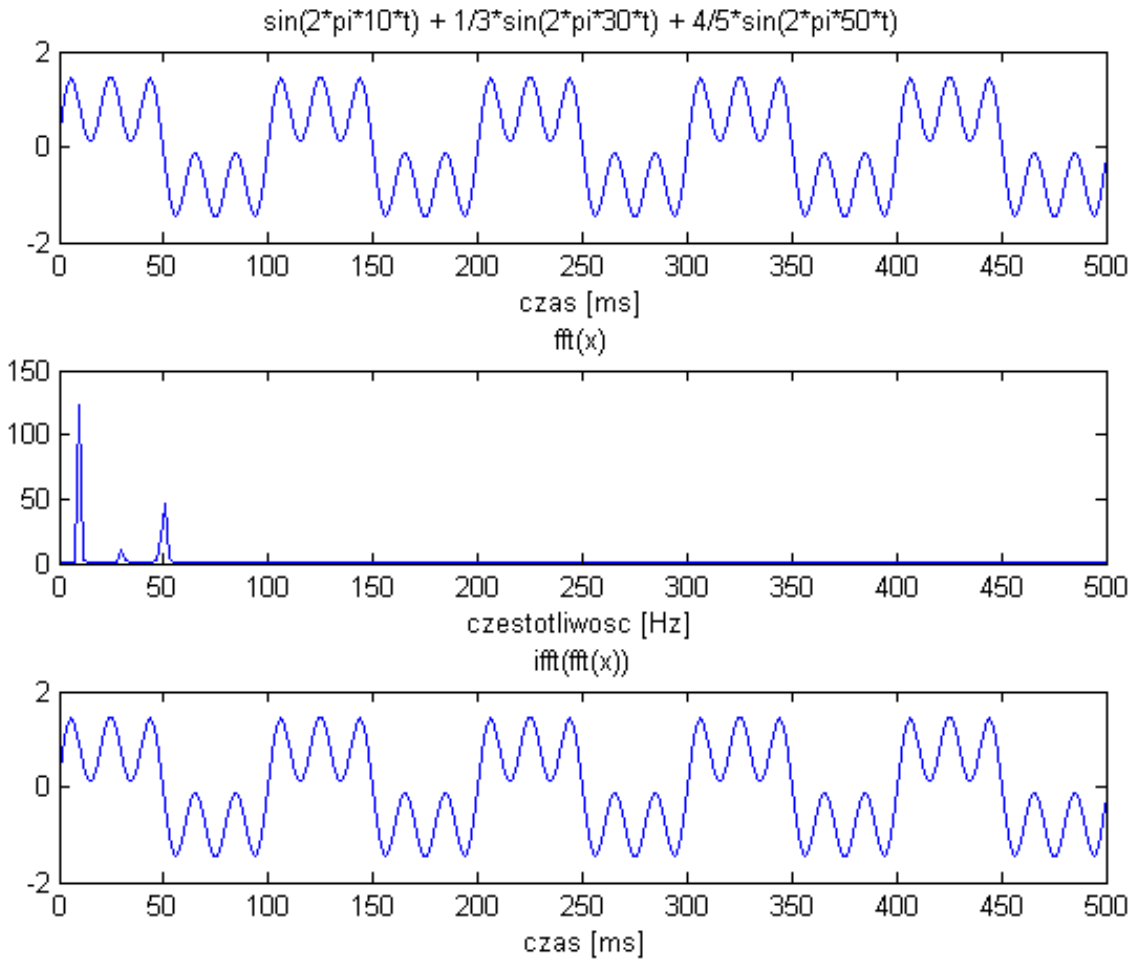
Przykład: transformacja sinusa



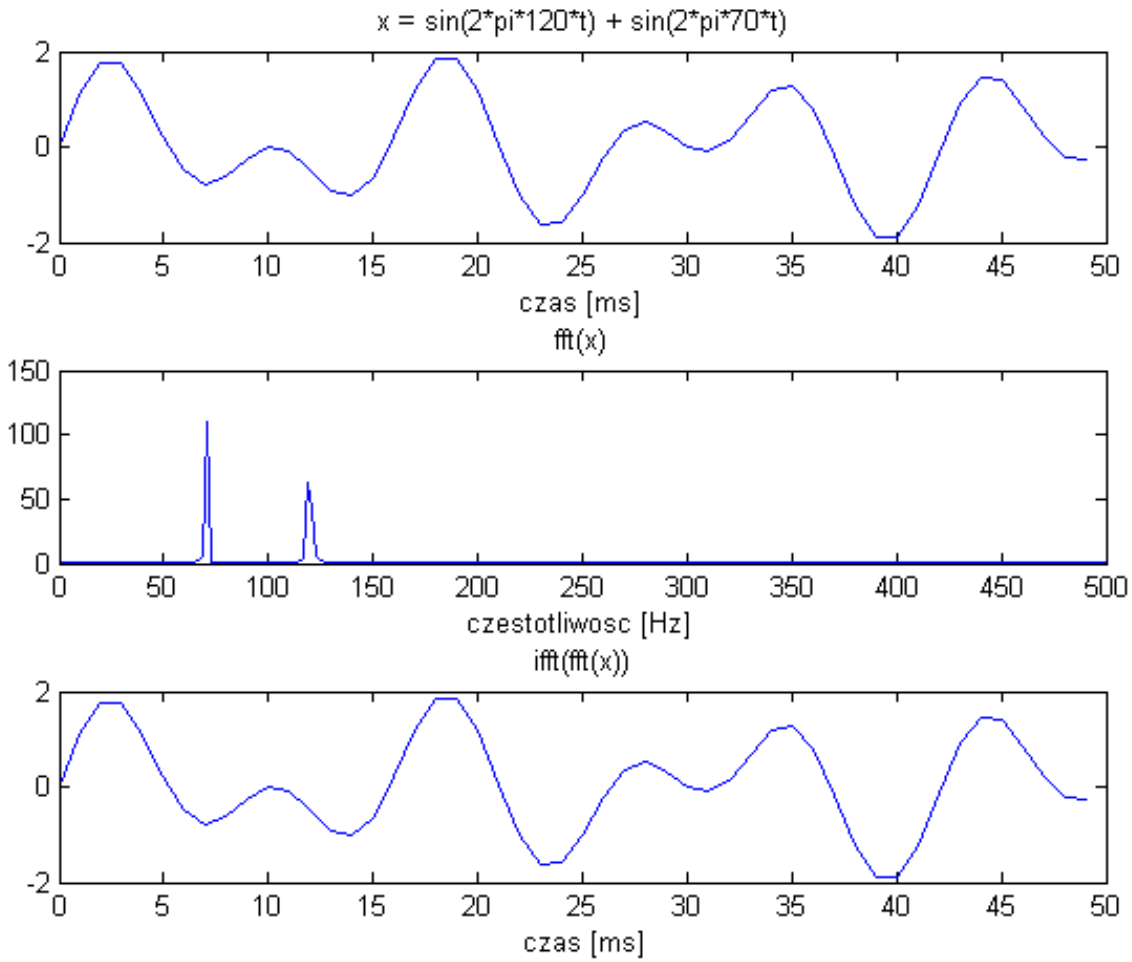
Przykład: transformacja sumy sinusów



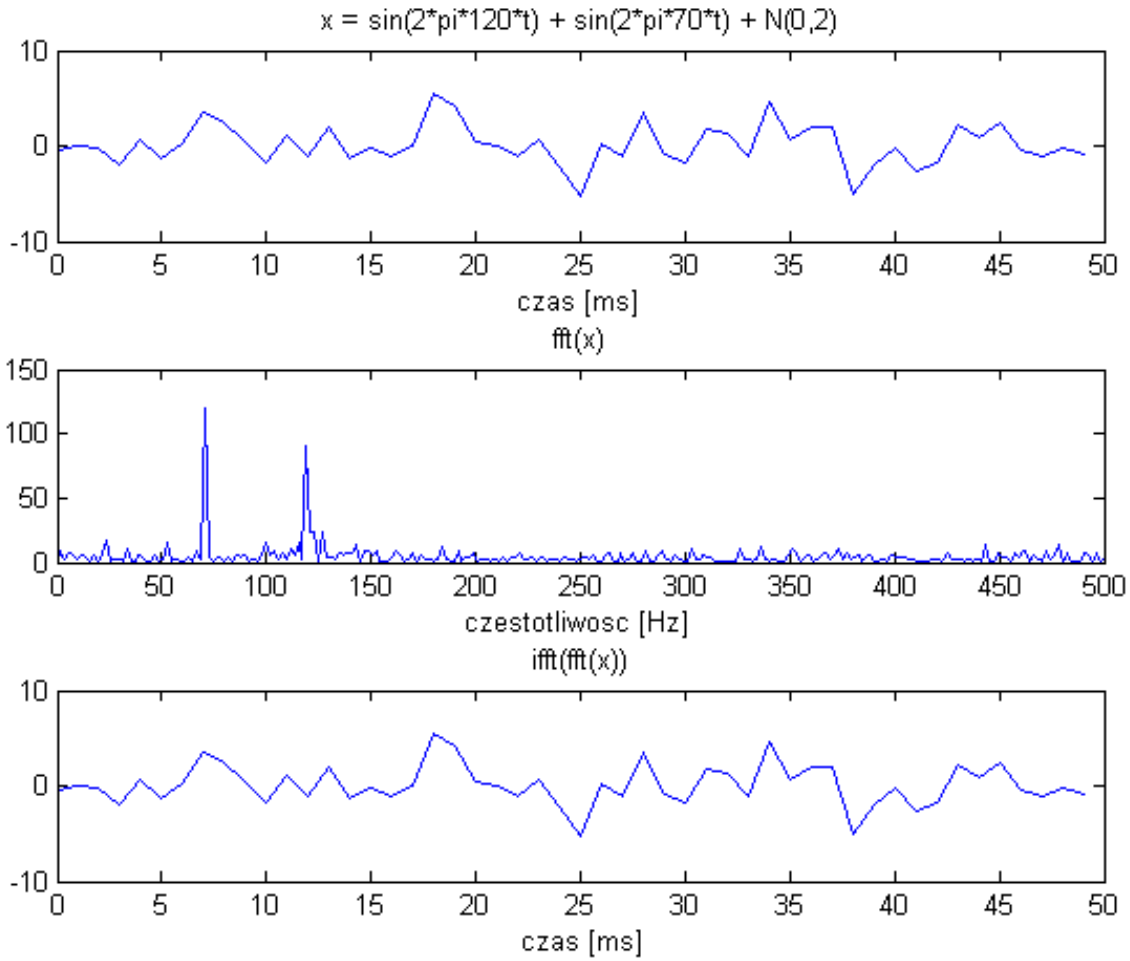
Przykład: transformacja sumy sinusów



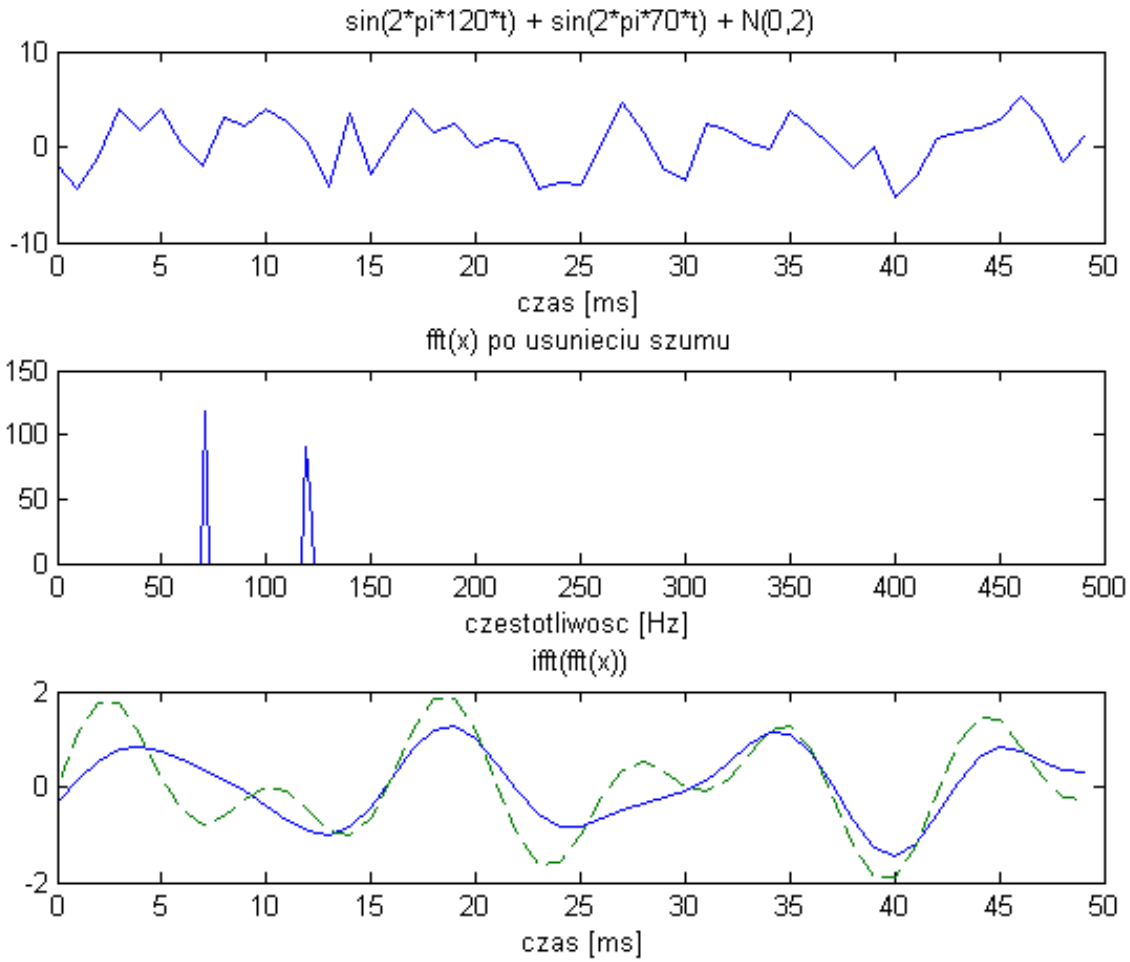
Przykład: transformacja sygnałów zaszumionych



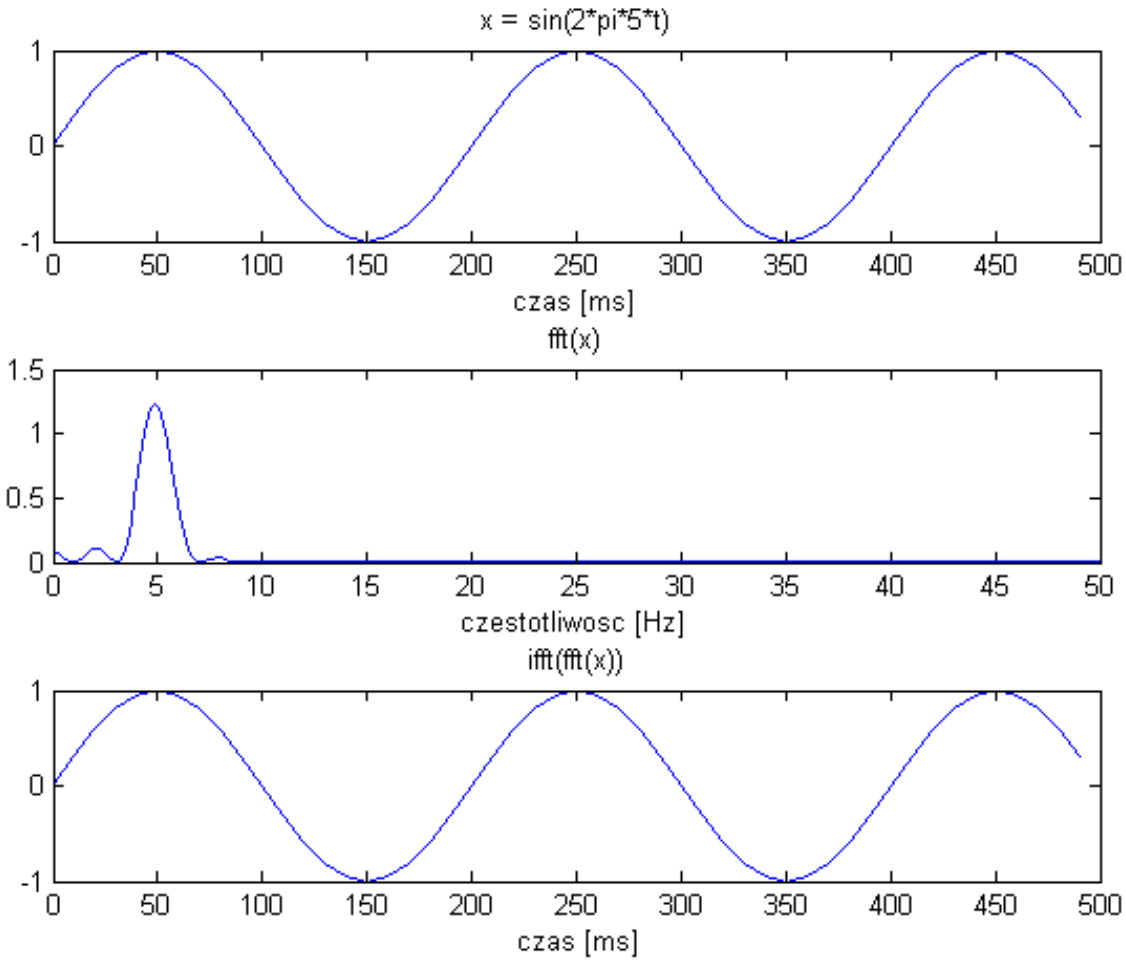
Przykład: transformacja sygnałów zaszumionych



Przykład: transformacja sygnałów zaszumionych



Problemy z transformatą dyskretną



Własności

- Liniowość: jeżeli $h(x) = af(x) + bg(x)$, to $\hat{h}(\xi) = a \cdot \hat{f}(\xi) + b \cdot \hat{g}(\xi)$
- Translacje: jeżeli $h(x) = f(x - x_0)$, to $\hat{h}(\xi) = e^{-2\pi i x_0 \xi} \hat{f}(\xi)$
- Modulacja: jeżeli $h(x) = e^{2\pi i x \xi_0} f(x)$, to $\hat{h}(\xi) = \hat{f}(\xi - \xi_0)$
- Skalowanie: jeżeli $h(x) = f(ax)$, to $\hat{h}(\xi) = \frac{1}{|a|} \hat{f}\left(\frac{\xi}{a}\right)$
  - dla $a = -1$ jeżeli $h(x) = f(-x)$, to $\hat{h}(\xi) = \hat{f}(-\xi)$
- Sprzężenie: jeżeli $h(x) = \overline{f(x)}$, to $\hat{h}(\xi) = \overline{\hat{f}(-\xi)}$
  - jeżeli f jest funkcją rzeczywistą, to $\hat{f}(-\xi) = \overline{\hat{f}(\xi)}$
- Tw. o splocie: jeżeli $h(x) = (f * g)(x)$, to $\hat{h}(\xi) = \hat{f}(\xi) \cdot \hat{g}(\xi)$

Zastosowania

- Przetwarzanie sygnałów
- Przetwarzanie obrazów, filtry
- Równania różniczkowe
- Obliczanie splotów
  - sumowanie zmiennych losowych
  - mnożenie wielomianów
  - mnożenie dużych liczb
- Kompresja danych (dyskretne transformaty kosinusowe) np. JPEG, MP3

FFT i CUFFT

Fast Fourier Transform

- Algorytm (rodzina algorytmów) do obliczania DFT i jej odwrotności
- Ma złożoność $O(nlog(n))$, opiera się na zasadzie "dziel i rządź"
- Najszybciej i najdokładniej działa, gdy długość danych podlegających transformacji jest potęgą pewnej małej liczby pierwszej, najlepiej dwójki
- Popularna biblioteka FFTW (Fastest Fourier Transform in the West) dla C
- CUFFT ma podobne API do FFTW
- NVIDIA *CUDA FFT library documentation*, February 2010

Własności CUFFT

- Transformaty sygnałów rzeczywistych i zespolonych 1D, 2D i 3D
- Wiele transformacji dowolnych wymiarów równolegle
- Transformaty 2D i 3D mogą przetwarzać sygnały w zakresie [2, 16384] w każdym wymiarze
- Transformaty 1D mogą przetwarzać do 8 milionów elementów
- Transformaty w miejscu i zewnętrzne
- Podwójna precyzja na odpowiednim sprzęcie (GT200 i późniejsze) – nowość w wersji 3.0
- Wsparcie wykonywania w strumieniach, możliwość asynchronicznego przesyłu danych

Typy danych CUFFT

- `typedef float cufftReal`
- `typedef double cufftDoubleReal`
- `typedef cuComplex cufftComplex`
- `typedef cuDoubleComplex cufftDoubleComplex`

CUFFT typy transformacji

Typy transformacji:

```
typedef enum cufftType_t {
CUFFT_R2C = 0x2a, // Real to complex (interleaved)
CUFFT_C2R = 0x2c, // Complex (interleaved) to real
CUFFT_C2C = 0x29, // Complex to complex, interleaved
CUFFT_D2Z = 0x6a, // Double to double-complex
CUFFT_Z2D = 0x6c, // Double-complex to double
CUFFT_Z2Z = 0x69 // Double-complex to double-complex
} cufftType;

#define CUFFT_FORWARD -1
#define CUFFT_INVERSE 1
```

1D real-to-complex

```
#define NX 256
#define BATCH 10

cufftHandle plan;
cufftComplex *data;
cudaMalloc((void**)&data, sizeof(cufftComplex)*(NX/2+1)*BATCH);

/* Create a 1D FFT plan. */
cufftPlan1d(&plan, NX, CUFFT_R2C, BATCH);

/* Use the CUFFT plan to transform the signal in place. */
cufftExecR2C(plan, (cufftReal*)data, data);

/* Destroy the CUFFT plan. */
cufftDestroy(plan);
cudaFree(data);
```

2D complex-to-real

```
#define NX 256
#define NY 128
cufftHandle plan;
cufftComplex *idata;
cufftReal *odata;

cudaMalloc((void**)&idata, sizeof(cufftComplex)*NX*NY);
cudaMalloc((void**)&odata, sizeof(cufftReal)*NX*NY);

/* Create a 2D FFT plan. */
cufftPlan2d(&plan, NX, NY, CUFFT_C2R);
/* Use the CUFFT plan to transform the signal out of place. */
cufftExecC2R(plan, idata, odata);

/* Destroy the CUFFT plan. */
cufftDestroy(plan);
cudaFree(idata); cudaFree(odata);
```

Wtyczki

- Matlab
- LabView
- Mathematica
- Python
- R

Wydajność

CUDA FFT, a FFTW

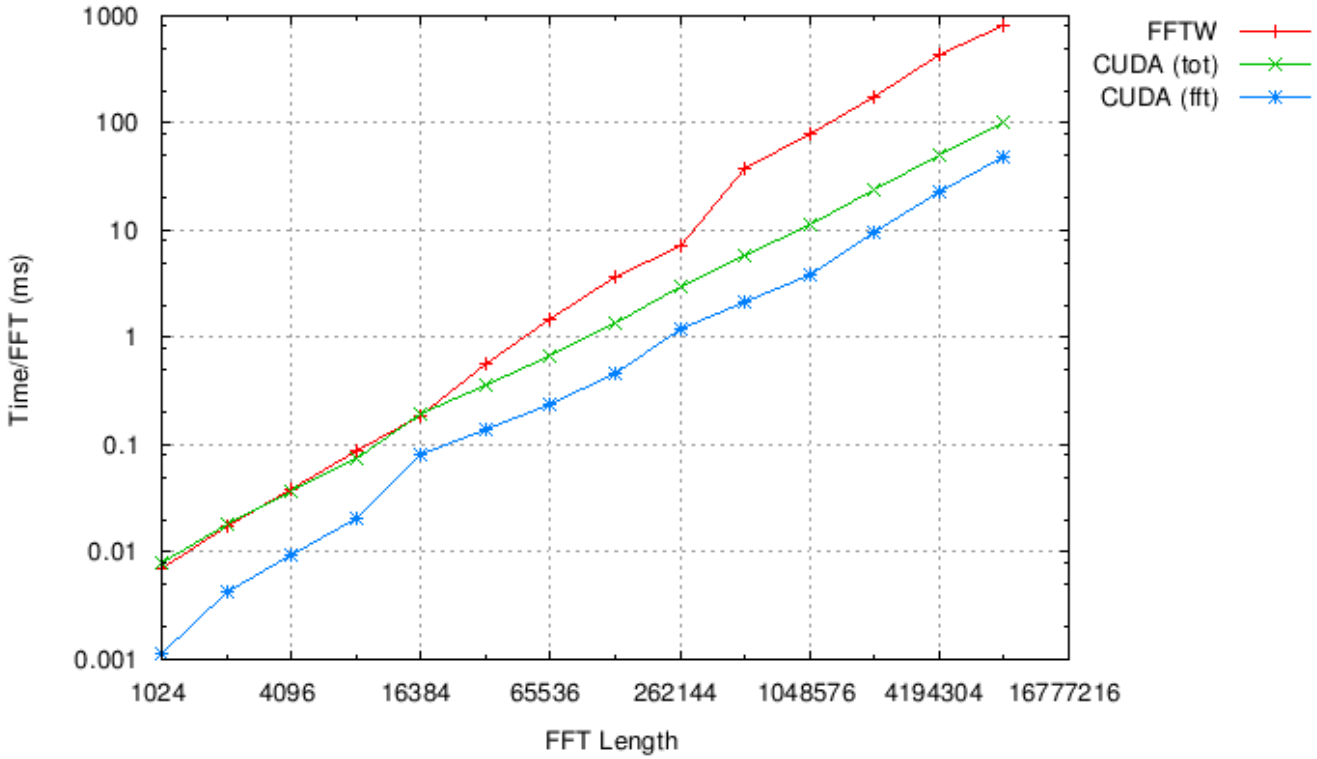
Paul Demorest, National Radio Astronomy Observatory, USA, sierpień 2007

- CPU: Intel Core 2 Quad, 2.4GHz
- GPU: NVIDIA GeForce 8800 GTX

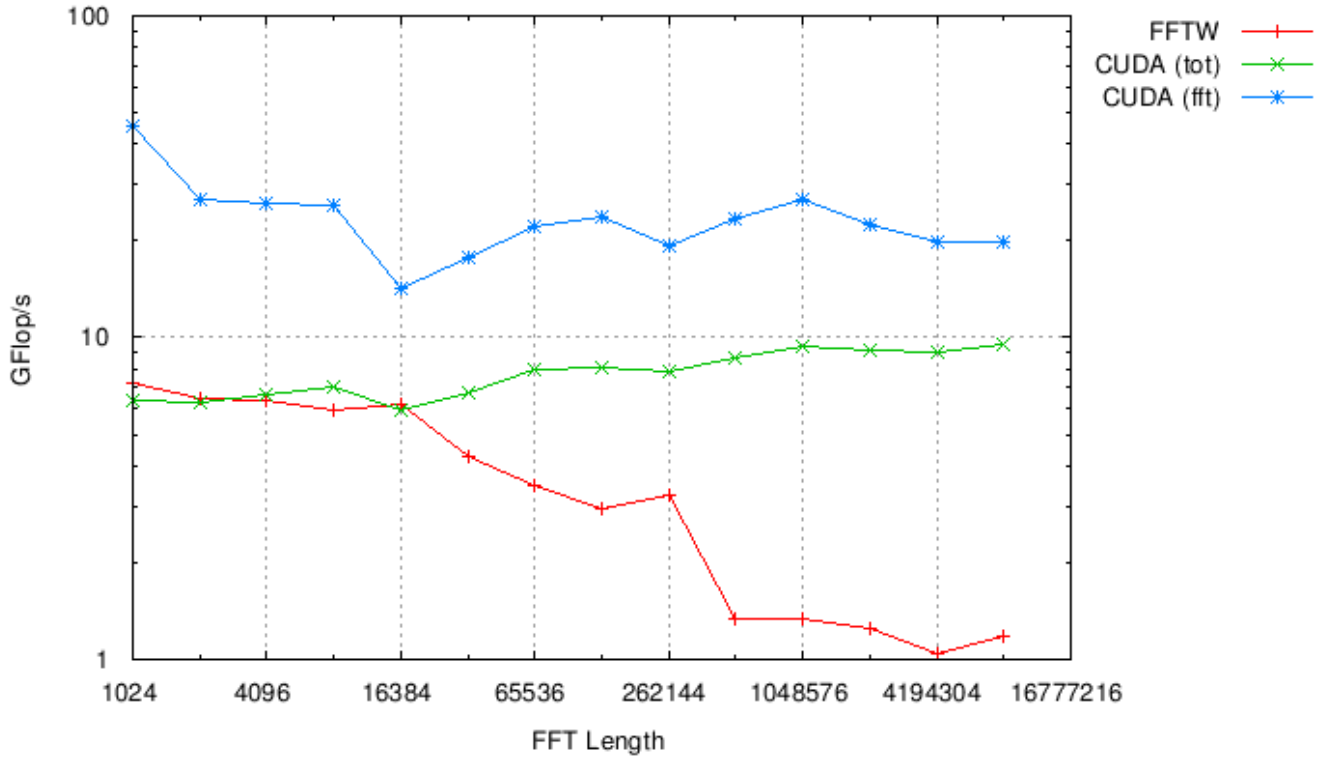
Procedura testowa, wszystkie transformacje "w miejscu"

1. $8M = 8 \cdot 10^6$ losowych float-ów (64MB).
2. Dane przesyłane do GPU (jeśli zachodzi taka potrzeba)
3. Dane dzielone na 8M/fftLen kawałków (chunks) i dla każdego uruchamiano wsadowe wywołanie FFTW/CUFFT
4. Wyniki przesyłane z powrotem do CPU

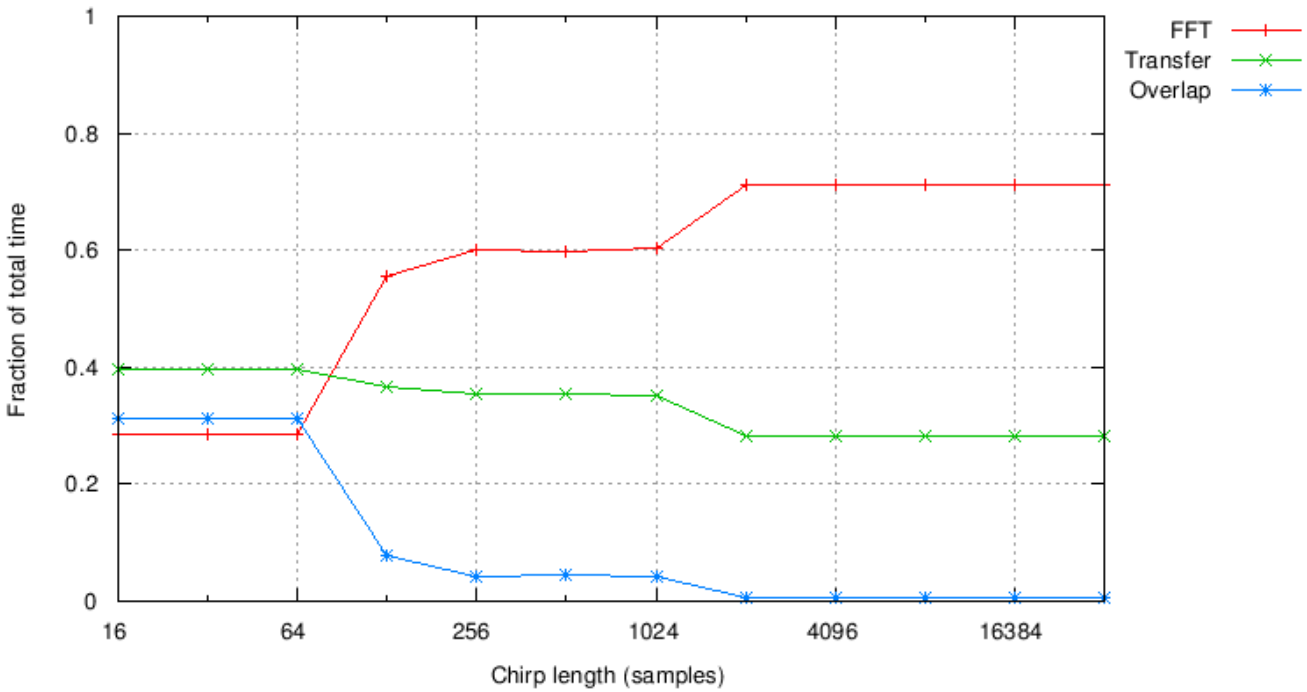
Czas/FFT


1000
100
10
1
0.1
0.01
0.001
Time/FFT (ms)
1024
4096
16384
65536
262144
1048576
4194304
16777216
FFT Length
FFTW
CUDA (tot)
CUDA (fft)

Gflops/s


100
10
1
GFlop/s
1024
4096
16384
65536
262144
1048576
4194304
16777216
FFT Length
FFTW
CUDA (tot)
CUDA (fft)

Wykorzystanie czasu



Podsumowanie

1. Dość wygodna biblioteka
2. Może pokonać najlepsze biblioteki na CPU, jeśli dane są dostatecznie duże
3. Wąskim gardłem są transfery pamięci
4. Można mieć nadzieję na poprawę szybkości

Dziękuję za uwagę!