# Fast Convolution and Fast Fourier Transform under Interval Uncertainty

Guoqing Liu[1] and Vladik Kreinovich[2]
[1] School of Sciences, Nanjing University of Technology,
Nanjing, Jiangsu 210009, P.R. China, guoqing@njut.edu.cn
[2]Dept. of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA, vladik@utep.edu

**Abstract**

Convolution $y(t) = \int a(t-s) \cdot x(s)\,ds$ is one of the main techniques in digital signal processing. A straightforward computation of the convolution $y(t)$ requires $O(n^2)$ steps, where $n$ is the number of observations $x(t_0), \ldots, x(t_{n-1})$. It is well known that by using the Fast Fourier Transform (FFT) algorithm, we can compute convolution much faster, with computation time $O(n \cdot \log(n))$.

In practice, we only know the signal $x(t)$ and the function $a(t)$ with uncertainty; sometimes, we know them with interval uncertainty, i.e., we know intervals $[\underline{x}(t), \overline{x}(t)]$ and $[\underline{a}(t), \overline{a}(t)]$ that contain the actual (unknown) functions $x(t)$ and $a(t)$. In such situations, it is desirable, for every $t$, to compute the range $[\underline{y}(t), \overline{y}(t)]$ of possible values of $y(t)$. Of course, it is possible to use straightforward interval computations to compute this range, i.e., replace every computational step in FFT by the corresponding operations of interval arithmetic. However, the resulting enclosure is too wide.

In this paper, we show how to provide asymptotically accurate ranges for $y(t)$ in time $O(n \cdot \log(n))$.

## 1 Formulation of the Problem

**Convolution is important.** The output $y(t)$ of an electronic device depends on its inputs $x(t)$. In many practical situations, the input signal $x(t)$ is reasonably small; as a result, we can safely ignore quadratic and higher order terms in the dependence of $y(t)$ on $x(t)$ and assume that the dependence is linear, i.e., that $y(t) = a_0(t) + \int a_1(t,s) \cdot x(s)\,ds$ for some coefficients $a_0(t)$ and $a_1(t,s)$.

In many interesting cases, the signal processing device does not change in time, in the sense that if we feed the same input again, we get the same output.

In precise terms, this means that for every time shift $t_0$, if we input the signal $x(t+t_0)$, we should get $y(t+t_0)$, i.e., we should have

$$a_0(t) + \int a_1(t,s)\cdot x(s+t_0)\,ds = a_0(t+t_0) + \int a_1(t+t_0,s)\cdot x(s)\,ds$$

for all functions $x(t)$ and for all real numbers $t$ and $t_0$. For $x(t) \equiv 0$, this requirement leads to $a_0(t) = a_0(t+t_0)$, i.e., to the conclusion that $a_0(t)$ is a constant. By subtracting this constant from $y(t)$, we get a simplified expression $y(t) = \int a_1(t,s)\cdot x(s)\,ds$ with

$$\int a_1(t,s)\cdot x(s+t_0)\,ds = \int a_1(t+t_0,s)\cdot x(s)\,ds.$$

By introducing a new variable $s+t_0$ in the first integral, we conclude that $\int a_1(t,s-t_0)\cdot x(s)\,ds = \int a_1(t+t_0,s)\cdot x(s)\,ds$ for all $t$ and all $x(s)$. In particular, for a pulse signal, i.e., for function $x(t)$ which is non-zero only in a small neighborhood of a point $s$, we conclude that $a_1(t,s-t_0) = a_1(t+t_0,s)$ for all $t$, $s$, and $t_0$.

For every $t'$ and $s$, we can take $t_0 = s$ and $t = t - s$, then we get $s - t_0 = 0$ and $a_1(t',s) = a_1(t+t_0,s) = a_1(t'-s,0)$. Thus, for $a(t) \stackrel{\text{def}}{=} a_1(t,0)$, we conclude that

$$y(t) = \int a(t-s)\cdot x(s)\,ds.$$

In digital signal processing, this formula is called a *convolution*; see, e.g., [10].

Similarly, it is reasonable to consider optical filters for processing 2-D or 3-D images $x(\vec{t})$. If we require that the result of filtering does not depend on the exact spatial location of the image, then we can conclude that these filters can also described as convolutions: $y(\vec{t}) = \int a(\vec{t}-\vec{s})\cdot x(\vec{s})\,d\vec{s}$ for some function $a(\vec{t})$.

Thus, if we want the compute to simulate the work of different electronic and/or optical devices such as filters, we must be able to compute one-dimensional and multi-dimensional convolutions.

**It is important to compute convolutions fast.** In computation, the input signal $x(t)$ is represented by its samples $x_0 = x(t_0), x_1 = x(t_1), \ldots, x_{n-1} = x(t_{n-1})$, usually measured at equally spaced moments of time $t_k = t_0 + k\cdot\Delta t$. Similarly, the input image $x(\vec{t})$ is usually represented by the intensity values at pixels forming a rectangular grid.

Based on this information, we can approximate the integral by the corresponding integral sum, and compute $n$ values $y_i \stackrel{\text{def}}{=} y(t_i)$ as

$$y_i = \sum_{i=0}^{n-1} b_{i-k}\cdot x_k,$$

where $b_i \stackrel{\text{def}}{=} a(i \cdot \Delta t) \cdot \Delta t$. This discrete convolution operation that transforms two sequences $b = (b_0, \ldots, b_{n-1})$ and $x = (x_0, \ldots, x_{n-1})$ into a new sequence $y = (y_0, \ldots, y_{n-1})$ is usually denoted by $*$:

$$y = b * x.$$

If we use this formula to compute the outputs $y_i$, then we need $n$ multiplications and $n$ additions to compute each of $n$ outputs, to the total of $O(n) \cdot O(n) = O(n^2)$ computational steps.

However, the number of samples $n$ in usually in thousands and millions (an image is usually several Megabytes); for such large $n$, $n^2$ steps require too much time. For example, for $n \approx 10^6$, $n^2$ steps would require $\approx 10^{12}$ steps, i.e., about 15 minutes on a standard Gigahertz computer.

**Fast Fourier Transform leads to fast computation of convolution.** It is well known that convolution can be computed much faster, in time $O(n \cdot \log(n))$, if we use the $O(n \cdot \log(n))$ Fast Fourier Transform (FFT) algorithm for computing Fourier transforms $\hat{x}(\omega) \stackrel{\text{def}}{=} \dfrac{1}{\sqrt{2\pi}} \cdot \int x(t) \cdot \exp(-\mathrm{i} \cdot t \cdot \omega)\, dt$; see, e.g., [1, 10] (Please notice that in this paper, we use mathematical notation i for $\sqrt{-1}$; in signal processing, $\sqrt{-1}$ is usually denoted by $j$, to avoid confusion with the current $i$.)

Namely, it is known that the Fourier transform of the convolution is equation to the product of the Fourier transforms: $\hat{y}(\omega) = \hat{a}(\omega) \cdot \hat{x}(\omega)$. Thus, to compute the convolution $y(t)$, we do the following:

- first, we apply FFT to $x(t)$ and $a(t)$ and compute $\hat{x}(\omega)$ and $\hat{a}(\omega)$;
- then, we multiply $\hat{x}(\omega)$ and $\hat{a}(\omega)$, thus computing $\hat{y}(\omega) = \hat{a}(\omega) \cdot \hat{x}(\omega)$;
- finally, we apply the inverse FFT to $\hat{y}(\omega)$ and compute $y(t) = \dfrac{1}{\sqrt{2\pi}} \cdot \int \hat{y}(\omega) \cdot \exp(\mathrm{i} \cdot t \cdot \omega)\, d\omega$.

To be more precise, when we have two sequences $x_i$ and $b_i$, we do the following:

- first, we apply discretized FFT to $x_i$ and compute the corresponding Fourier coefficients as $X_j = \dfrac{1}{\sqrt{n}} \cdot \sum\limits_{k=0}^{n-1} x_k \cdot \exp\left(-\mathrm{i} \cdot 2\pi \cdot \dfrac{j \cdot k}{n}\right)$; similarly, we compute the Fourier coefficients $B_j$ of the sequence $b_i$;
- then, we multiply $X_j$ and $B_j$, thus computing $Y_j = A_j \cdot B_j$;
- finally, we apply the inverse FFT to $Y_j$ and compute

$$y_k = \frac{1}{\sqrt{n}} \cdot \sum_{k=0}^{n-1} Y_j \cdot \exp\left(\mathrm{i} \cdot 2\pi \cdot \frac{j \cdot k}{n}\right).$$

Similarly, for every natural number $d$, convolution of $d$-dimensional functions $x(\vec{t})$ can be computed by using multi-dimensional Fast Fourier Transform $\hat{x}(\vec{\omega}) \stackrel{\rm def}{=} \frac{1}{(\sqrt{2\pi})^d} \cdot \int x(\vec{t}) \cdot \exp(-{\rm i} \cdot (\vec{t} \cdot \vec{\omega}))\, d\vec{t}$ – or, to be more precise, the discretized multi-dimensional FFT.

**In practice, we often have interval uncertainty.** In practice, we only know the signal $x(t)$ and the function $a(t)$ with uncertainty; in other words, we only know the approximate (measured) values $\widetilde{x}(t)$ and $\widetilde{a}(t)$ of these functions. In this case, the convolution $\widetilde{y}(t)$ of these approximate functions is also only an approximation to the desired convolution $y(t)$ of the (unknown) actual values $x(t)$ and $a(t)$.

Usually, we know the upper bounds $\Delta_x(t)$ and $\Delta_a(t)$ on the deviations $\Delta x(t) \stackrel{\rm def}{=} x(t) - \widetilde{x}(t)$ and $\Delta a(t) \stackrel{\rm def}{=} a(t) - \widetilde{a}(t)$. In some cases, we also know the probability of different deviations, but often, these upper bounds is all we know. In such cases, we only know the intervals $[\widetilde{x}(t) - \Delta_x(t), \widetilde{x}(t) + \Delta_x(t)]$ and $[\widetilde{a}(t) - \Delta_a(t), \widetilde{a}(t) + \Delta_a(t)]$ that contain the actual (unknown) functions $x(t)$ and $a(t)$.

In terms of sample values, we know the intervals $\mathbf{x}_i = [\widetilde{x}_i - \Delta_{xi}, \widetilde{x}_i + \Delta_{xi}]$ and $\mathbf{b}_i = [\widetilde{b}_i - \Delta_{bi}, \widetilde{b}_i + \Delta_{bi}]$ that contain the actual (unknown) values of $x_i$ and $b_i$.

In such situations, it is desirable, for every $t$, to compute the range of possible values of $y(t)$, i.e., to find the upper bounds $\Delta y(t)$ such that for every moment $t$, the actual value of the convolution $y(t)$ always lies within the interval $[\widetilde{y}(t) - \Delta_y(t), \widetilde{y}(t) + \Delta_y(t)]$. In terms of sample values, we need to find intervals $\mathbf{y}_i = [\widetilde{y}_i - \Delta_{yi}, \widetilde{y}_i + \Delta_{yi}]$ that contain the actual (unknown) values $y_i$.

**It is possible to compute convolution under interval uncertainty in time $O(n^2)$.** In terms of sample values, computing convolution means computing the values $y_i = \sum\limits_{k=0}^{n-1} b_{i-k} \cdot x_k$. For each $i$, the expression for $y_i$ is a single-use expression (SUE). Thus (see, e.g., [4, 6]), we can compute the exact range $\mathbf{y}_i$ for $y_i$ by replacing each arithmetic operation in this expression with the corresponding operation of interval arithmetic:

$$\mathbf{y}_i = \sum_{k=0}^{n-1} \mathbf{b}_{i-k} \cdot \mathbf{x}_k \qquad (1).$$

The problem with this computation is that it requires $O(n)$ computational steps for each of $n$ values $i$, to the total of $O(n^2)$ steps – and we already know that this is too long. It is desirable to design faster algorithms for computing convolution under interval uncertainty.

**Straightforward interval version of FFT leads to too wide enclosures.** In principle, it is possible to use straightforward interval computations to compute this range, i.e., replace every computational step in FFT by the corresponding operations of interval arithmetic. However, as noticed already in the pioneering paper by [2], the resulting enclosure is too wide.

Let us explain this on the simple example of $n = 2$ data points $x_0$ and $x_1$ known with interval uncertainty: $\mathbf{x}_0 = \mathbf{x}_1 = [-\Delta, \Delta]$. In the simplest case when the input does not change $(y(t) = x(t))$, i.e., when $A_i = 1$, the FFT algorithm requires that we first compute the FFT $X_j$ of the sequence $x_i$, then multiply $X_j$ by $A_j = 1$ (i.e., do not change), and then apply inverse FFT to the values $Y_j = X_j$. When we know $x_i$ exactly, applying inverse FFT to the FFT values returns the exact same values $x_i$. However, for intervals, we get excess width.

Indeed, for $n = 2$, discrete FFT takes the form $X_0 = \dfrac{x_0 + x_1}{\sqrt{2}}$ and $X_1 = \dfrac{x_0 - x_1}{\sqrt{2}}$, and the inverse FFT takes a similar form $x_0 = \dfrac{X_0 + X_1}{\sqrt{2}}$ and $x_1 = \dfrac{X_0 - X_1}{\sqrt{2}}$. For interval data, we thus get

$$\mathbf{X}_0 = \frac{\mathbf{x}_0 + \mathbf{x}_1}{\sqrt{2}} = \frac{[-\Delta, \Delta] + [-\Delta, \Delta]}{\sqrt{2}} = [-\sqrt{2} \cdot \Delta, \sqrt{2} \cdot \Delta]$$

and $\mathbf{X}_1 = \dfrac{\mathbf{x}_0 - \mathbf{x}_1}{\sqrt{2}} = [-\sqrt{2} \cdot \Delta, \sqrt{2} \cdot \Delta]$, after which we get $\mathbf{x}_0 = \dfrac{\mathbf{X}_0 + \mathbf{X}_1}{\sqrt{2}} = [-2 \cdot \Delta, 2 \cdot \Delta]$ and similarly, $\mathbf{x}_1 = [-2 \cdot \Delta, 2 \cdot \Delta]$. Since straightforward interval computations ignore the dependence between $X_0$ and $X_1$, we get intervals which are twice wider than the actual range.

**What we do in this paper.** Examples like the one above lead to an impression that doing convolution via FFT is impossible without really inflating the intervals.

In this paper, we show, that, contrary to this impression, it is possible to compute convolution via FFT (i.e., fast) without generating significant interval inflation.

## 2 Main Result: Fast Convolution under Interval Uncertainty

**We will use Rump's circular arithmetic.** In this paper, instead of the standard interval arithmetic, we will use Rump's circular arithmetic (see, e.g., [8]) that provides exact range for addition and subtraction and asymptotically precise range for multiplication.

In the circular arithmetic,

$$[\widetilde{a}-\Delta_a,\widetilde{a}+\Delta_a]+[\widetilde{b}-\Delta_b,\widetilde{b}+\Delta_b]=[(\widetilde{a}+\widetilde{b})-(\Delta_a+\Delta_b),(\widetilde{a}+\widetilde{b})+(\Delta_a+\Delta_b)]$$

and

$$[\widetilde{a}-\Delta_a,\widetilde{a}+\Delta_a]\cdot[\widetilde{b}-\Delta_b,\widetilde{b}+\Delta_b]=[\widetilde{a}\cdot\widetilde{b}-\Delta,\widetilde{a}\cdot\widetilde{b}+\Delta],$$

where

$$\Delta=|a|\cdot\Delta_b+\Delta_a\cdot|b|+\Delta_a\cdot\Delta_b.$$

*Comments.*

- In these two operations, the midpoint of each resulting interval is equal to the result of applying the same arithmetic operation to the corresponding midpoints. It is known that the same property holds for any operation or any sequence of operations of circular arithmetic.
- In the above formulas, we did not take rounding errors into account because in the signal processing applications, rounding errors are usually negligible in comparison with the measurement errors. If necessary, rounding errors can be taken into account by introducing appropriate roundings [8] – as it is usually done in interval computations (see, e.g., [6]).

**Analysis of the problem.** Applying the formulas for circular arithmetic to the expression (1), we conclude that $\mathbf{y}_i=[\widetilde{y}_i-\Delta_{yi},\widetilde{y}_i+\Delta_{yi}]$, where

$$\Delta_{yi}=\sum_{k=0}^{n-1}|\widetilde{b}_{i-k}|\cdot\Delta_{xk}+\sum_{k=0}^{n-1}\Delta b_{i-k}\cdot|x_k|+\sum_{k=0}^{n-1}\Delta b_{i-k}\cdot\Delta_{xk}.$$

Thus, once we have the sequences $|b|=(|b_0|,\ldots,|b_{n-1}|)$, $|x|=(|x_0|,\ldots,|x_{n-1}|)$ $\Delta_b=(\Delta_{b0},\ldots,\Delta_{b,n-1})$, and $\Delta_x=(\Delta_{x0},\ldots,\Delta_{x,n-1})$, we can compute the desired sequence $\Delta y=(\Delta_{y0},\ldots,\Delta_{y,n-1})$ as the sum of three convolutions:

$$\Delta y=|b|*\Delta_x+\Delta_b*|x|+\Delta_b*\Delta*x. \tag{2}$$

Since by using FFT, we can compute each of these three convolutions fast (in time $O(n\cdot\log(n))$), we thus arrive at the following fast algorithm for computing convolution under interval uncertainty.

**Fast algorithm for computing convolution under interval uncertainty.** Suppose that we are given the intervals $[\widetilde{b}_i-\Delta_{bi},\widetilde{b}_i+\Delta_{bi}]$ and $[\widetilde{x}_i-\Delta_{xi},\widetilde{x}_i+\Delta_{xi}]$. Then, to compute the (asymptotically exact enclosure) $[\widetilde{y}_i-\Delta_{yi},\widetilde{y}_i+\Delta_{yi}]$, we do the following:

- first, we use FFT-based convolution algorithm to compute $\widetilde{y}=\widetilde{b}*\widetilde{x}$;

- then, we use FFT-based convolution algorithm to compute three auxiliary convolutions $|b| * \Delta_x$, $\Delta_b * |x|$, and $\Delta_b * \Delta * x$;

- finally, we add the resulting three sequences and compute $\Delta_y$ by using the formula (2).

Since each FFT-based convolution requires $O(n \cdot \log(n))$ steps, we thus arrive at the $O(n \cdot \log(n))$ algorithm for computing convolution under interval uncertainty.

*Comment.* The same algorithm works in multi-dimensional case as well.

**This algorithm can be made even faster.** In the above algorithm, we need four convolutions of numerical sequences to compute a convolution of two interval-valued sequences. It turns out that we can further speed up this computation because it is possible to use only 3 convolutions instead of 4.

Namely, since

$$(|b| + \Delta_b) * (|x| + \Delta_x) = |b| * |x| + |b| * \Delta_x + \Delta_b * |x| + \Delta_b * \Delta * x,$$

we can compute $\Delta_y$ as

$$\Delta_y = (|b| + \Delta_b) * (|x| + \Delta_x) - |b| * |x|. \tag{3}$$

Thus, we arrive at the following algorithm for computing convolution under interval uncertainty:

- first, we use FFT-based convolution algorithm to compute $\widetilde{y} = \widetilde{b} * \widetilde{x}$;

- then, we use FFT-based convolution algorithm to compute two auxiliary convolutions $(|b| + \Delta_b) * (|x| + \Delta_x)$ and $|b| * |x|$;

- finally, we subtract the resulting sequences and compute $\Delta_y$ by using the formula (3).

*Comment.* This possibility to reduce the number of underlying numerical operations from 4 to 3 is similar to the situation with standard interval multiplication

$$[\underline{a}, \overline{a}] \cdot [\underline{b}, \overline{b}] = [\min(\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b}), \max(\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b})].$$

In this situation,

- we seem to need 4 multiplication of numbers to compute the product of two intervals, but

- in reality, three multiplications are sufficient [5] (see also [3]).

## 3 Auxiliary Result: Fast Fourier Transform under Interval Uncertainty

**Need for FFT under interval uncertainty.** In the above text, we considered FFT as a technique to compute convolution fast. However, in many practical problems, we actually need to compute the Fourier transform $\hat{X}(\vec{\omega})$ of a given signal $x(\vec{t})$.

For example, in image processing, it is known that in the far-field (Fraunhofer) approximation, the observed signal is actually equal to the Fourier transform of the desired image; see, e.g., [7, 9, 11, 12]. So, if we want to reconstruct the original image, we must apply the inverse Fourier transform to the measurement results $x_r(\vec{t}) + \mathrm{i} \cdot x_i(\vec{t})$, and find the values

$$\hat{X}_r(\vec{\omega}) = \frac{1}{(\sqrt{2\pi})^d} \cdot \left( \int x_r(\vec{t}) \cdot \cos(\vec{\omega} \cdot \vec{t})\, d\vec{t} - \int x_i(\vec{t}) \cdot \sin(\vec{\omega} \cdot \vec{t})\, d\vec{t} \right); \quad (4)$$

$$\hat{X}_i(\vec{\omega}) = \frac{1}{(\sqrt{2\pi})^d} \cdot \left( \int x_r(\vec{t}) \cdot \sin(\vec{\omega} \cdot \vec{t})\, d\vec{t} + \int x_i(\vec{t}) \cdot \cos(\vec{\omega} \cdot \vec{t})\, d\vec{t} \right). \quad (5)$$

In real life, we may only know the real and imaginary part of the signal with interval uncertainty, i.e., we only know the intervals $[\widetilde{x}_r(\vec{t}) - \Delta_r(\vec{t}), \widetilde{x}_r(\vec{t}) + \Delta_r(\vec{t})]$ and $[\widetilde{x}_i(\vec{t}) - \Delta_i(\vec{t}), \widetilde{x}_i(\vec{t}) + \Delta_i(\vec{t})]$ that contain the actual (unknown) values of $x_r(\vec{t})$ and $x_i(\vec{t})$. In such situations, it is desirable to find, for every $\vec{\omega}$, the intervals of possible values of $\hat{X}_r(\vec{\omega})$ and $\hat{X}_i(\vec{\omega})$.

We will show how this can be done under the assumption that discretization error is much smaller than the measurement error $\Delta_r(\vec{t})$ and $\Delta_i(\vec{t})$ and thus, we can safely assume that we know the values of the signal $\widetilde{x}_r(\vec{t})$ and $\widetilde{x}_i(\vec{t})$ for all $\vec{t}$.

**What was known.** The solution to this problem is known for the case when all the measurements have the same measurement error, i.e., when $\Delta_r(\vec{t})$ and $\Delta_i(\vec{t})$ do not depend on $\vec{t}$. For this case, the solution is given in [2]. In this paper, we extend this solution to the general case.

**Analysis of the problem.** Formulas (4) and (5) are linear in terms of the unknown $x_r(\vec{t})$ and $x_i(\vec{t})$. Thus, to describe the range of the corresponding expressions, let us recall how to estimate the range of a general linear function under interval constraints.

In general, every value $x_j$ from the interval $[\widetilde{x}_j - \Delta_j, \widetilde{x}_j + \Delta_j]$ can be described as $x_j = \widetilde{x}_j + \Delta x_j$, where $|\Delta x_j| \le \Delta_j$. In terms of $\widetilde{x}_j$ and $\Delta x_j$, the value of a (general) linear function $y = a_0 + \sum\limits_{j=1}^N a_j \cdot x_j$ takes the form $\widetilde{y} + \Delta y$, where

$$\widetilde{y} \stackrel{\text{def}}{=} a_0 + \sum_{j=1}^N a_j \cdot \widetilde{x}_j \quad (6)$$

and $\Delta y = \sum\limits_{j=1}^{N} a_j \cdot \Delta x_j$. The largest possible value of $\Delta y$ for $\Delta x_j \in [-\Delta_j, \Delta_j]$ is attained when $\Delta x_j = \Delta_j$ for $a_j \geq 0$ and when $\Delta x_j = -\Delta_j$ for $a_j \leq 0$. In both cases, the largest value $\Delta$ of $\Delta y$ is equal to

$$\Delta = \sum_{j=1}^{N} |a_j| \cdot \Delta_j. \tag{7}$$

Thus, the range of a linear function $y = a_0 + \sum\limits_{j=1}^{N} a_j \cdot x_j$ under interval uncertainty $x_j \in [\widetilde{x}_j - \Delta_j, \widetilde{x}_j + \Delta_j]$ is equal to $[\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.

In particular, for $\hat{X}_r(\vec{\omega})$, the desired range is equal to

$$[\tilde{\hat{X}}_r(\vec{\omega}) - \delta_r(\vec{\omega}), \tilde{\hat{X}}_r(\vec{\omega}) + \delta_r(\vec{\omega})],$$

where $\tilde{\hat{X}}_r(\vec{\omega})$ is the real part of the (easy-to-compute) Fourier transform of the approximate function $\widetilde{x}_r(\vec{t}) + \mathrm{i} \cdot \widetilde{x}_i(\vec{t})$ and

$$\delta_r(\vec{\omega}) = \frac{1}{(\sqrt{2\pi})^d} \cdot \left( \int \Delta_r(\vec{t}) \cdot |\cos(\vec{\omega} \cdot \vec{t})| \, d\vec{t} + \int \Delta_i(\vec{t}) \cdot |\sin(\vec{\omega} \cdot \vec{t})| \, d\vec{t} \right). \tag{8}$$

Similarly, for $\hat{X}_i(\vec{\omega})$, the desired range is equal to $[\tilde{\hat{X}}_i(\vec{\omega}) - \delta_i(\vec{\omega}), \tilde{\hat{X}}_i(\vec{\omega}) + \delta_i(\vec{\omega})]$, where $\tilde{\hat{X}}_i(\vec{\omega})$ is the imaginary part of the (easy-to-compute) Fourier transform of the approximate function $\widetilde{x}_r(\vec{t}) + \mathrm{i} \cdot \widetilde{x}_i(\vec{t})$ and

$$\delta_i(\vec{\omega}) = \frac{1}{(\sqrt{2\pi})^d} \cdot \left( \int \Delta_r(\vec{t}) \cdot |\sin(\vec{\omega} \cdot \vec{t})| \, d\vec{t} + \int \Delta_i(\vec{t}) \cdot |\cos(\vec{\omega} \cdot \vec{t})| \, d\vec{t} \right). \tag{9}$$

Thus, to be able to find these ranges, we must be able to compute the integrals

$$I_{rc} \stackrel{\text{def}}{=} \int \Delta_r(\vec{t}) \cdot |\cos(\vec{\omega} \cdot \vec{t})| \, d\vec{t}, \quad I_{ic} \stackrel{\text{def}}{=} \int \Delta_i(\vec{t}) \cdot |\cos(\vec{\omega} \cdot \vec{t})| \, d\vec{t},$$

$$I_{rs} \stackrel{\text{def}}{=} \int \Delta(\vec{t}) \cdot |\cos(\vec{\omega} \cdot \vec{t})| \, d\vec{t}, \quad I_{is} \stackrel{\text{def}}{=} \int \Delta_i(\vec{t}) \cdot |\sin(\vec{\omega} \cdot \vec{t})| \, d\vec{t}.$$

Let us consider the first of these integrals (the other three can be computed similarly). By definition, $\vec{\omega} \cdot \vec{t} = \omega_1 \cdot t_1 + \ldots + \omega_d \cdot t_d$; thus,

$$I_{rc}(\omega_1, \ldots, \omega_d) = \int \Delta_r(t_1, \ldots, t_d) \cdot |\cos(\omega_1 \cdot t_1 + \ldots + \omega_d \cdot t_d)| \, dt_1 \ldots dt_d.$$

By introducing the new variables $W_i \stackrel{\text{def}}{=} \ln(\omega_i)$ and $T_i \stackrel{\text{def}}{=} -\ln(t_i)$, for which $\omega_i = e^{W_i}$ and $t_i = e^{-T_i}$, we conclude that $dt_i = -e^{-T_i} \cdot dT_i$ and hence, the desired integral takes the form

$$I_{rc}(\omega_1, \ldots, \omega_d) =$$

$$(-1)^d\cdot\int \Delta_r(e^{-T_1},\ldots,e^{-T_d})\cdot e^{-(T_1+\ldots+T_d)}\cdot|\cos(e^{W_1-T_1}+\ldots+e^{W_d-T_d})|\,dT_1\ldots dT_d.$$

Thus,

$$I_{rc}(\omega_1,\ldots,\omega_d)=(-1)^d\cdot F_{rc}(\ln(\omega_1),\ldots,\ln(\omega_d)), \qquad (10)$$

where $F_{rc}(W_1,\ldots,W_d)$ is a convolution of the following two functions:

$$f_r(T_1,\ldots,T_d)=\Delta_r(e^{-T_1},\ldots,e^{-T_d})\cdot e^{-(T_1+\ldots+T_d)} \qquad (11)$$

and

$$g_c(T_1,\ldots,T_d)=|\cos(e^{T_1}+\ldots+e^{T_d})|. \qquad (12)$$

To compute the other three integrals, we also need

$$f_i(T_1,\ldots,T_d)=\Delta_i(e^{-T_1},\ldots,e^{-T_d})\cdot e^{-(T_1+\ldots+T_d)} \qquad (13)$$

and

$$g_s(T_1,\ldots,T_d)=|\sin(e^{T_1}+\ldots+e^{T_d})|. \qquad (14)$$

*Comment.* This idea works only for $\omega_i>0$ and $t_i>0$; to cover the entire integral, we must consider all $2^d$ (i.e., 4 or 8) orthants.

We know that convolution can be computed fast. Thus, we arrive at the following fact algorithm for computing Fourier transform under interval uncertainty.

**Resulting algorithm.** Suppose that we are given the intervals

$$[\widetilde{x}_r(\vec{t})-\Delta_r(\vec{t}),\widetilde{x}_r(\vec{t})+\Delta_r(\vec{t}] \text{ and } [\widetilde{x}_i(\vec{t})-\Delta_i(\vec{t}),\widetilde{x}_i(\vec{t})+\Delta_i(\vec{t})].$$

Then, to compute the ranges $[\widetilde{\widetilde{X}}_r(\vec{\omega})-\delta_r(\vec{\omega}),\widetilde{\widetilde{X}}_r(\vec{\omega})+\delta_r(\vec{\omega})]$ of the real and imaginary parts of the Fourier transform, we do the following:

- first, we apply FFT to the approximate function $\widetilde{x}_r(\vec{t})+\mathrm{i}\cdot\widetilde{x}_i(\vec{t})$ and compute $\widetilde{\widetilde{X}}_r(\vec{\omega})$ and $\widetilde{\widetilde{X}}_r(\vec{\omega})$;
- then, we use FFT-based convolution algorithm to compute the convolutions $F_{rc}=f_r*g_c$, $F_{rs}=f_r*g_s$, $F_{ic}=f_i*g_c$, and $F_{is}=f_i*g_s$;
- re-scale these functions by computing

$$I_{rc}(\omega_1,\ldots,\omega_d)=(-1)^d\cdot F_{rc}(\ln(\omega_1),\ldots,\ln(\omega_d)),$$
$$I_{rs}(\omega_1,\ldots,\omega_d)=(-1)^d\cdot F_{rs}(\ln(\omega_1),\ldots,\ln(\omega_d)),$$
$$I_{ic}(\omega_1,\ldots,\omega_d)=(-1)^d\cdot F_{ic}(\ln(\omega_1),\ldots,\ln(\omega_d)),$$
$$I_{is}(\omega_1,\ldots,\omega_d)=(-1)^d\cdot F_{is}(\ln(\omega_1),\ldots,\ln(\omega_d));$$

- finally, we compute

$$\delta_r(\vec{\omega}) = \frac{1}{(\sqrt{2\pi})^d} \cdot (I_{rc}(\vec{\omega}) + I_{is}(\vec{\omega})); \quad \delta_i(\vec{\omega}) = \frac{1}{(\sqrt{2\pi})^d} \cdot (I_{rs}(\vec{\omega}) + I_{ic}(\vec{\omega})).$$

Since we are only using FFT or FFT-based convolution, we thus arrive at a fast algorithm for computing Fourier Transform under interval uncertainty.

**Acknowledgments.** This work was done during Guoqing Liu's visit to the USA supported by the Jiangsu government scholarship. It was also supported in part by NASA under cooperative agreement NCC5-209, NSF grants EAR-0225670 and DMS-0532645, Army Research Lab grant DATM-05-02-C-0046, Star Award from the University of Texas System, and Texas Department of Transportation grant No. 0-5453.

The authors are very thankful to Andreas Griewank for his valuable advice.

## References

[1] Th. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, 2001.

[2] J. Garloff, "Zur intervallmässigen Durchführung der schnellen Fourier-Transformation", *ZAMM*, 1980, Vol. 60, pp. T291–T292.

[3] C. Hamzo and V. Kreinovich, "On Average Bit Complexity of Interval Arithmetic", *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 1999, Vol. 68, pp. 153–156.

[4] E. Hansen, "Sharpness in interval computations", *Reliable Computing*, 1997, Vol. 3, pp. 7–29.

[5] G. Heindl, *An improved algorithm for computing the product of two machine intervals*, Interner Bericht IAGMPI- 9304, Fachbereich Mathematik, Gesamthochschule Wuppertal, 1993.

[6] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, London, 2001.

[7] J. R. Meyer-Arendt, *Introduction to Classical and Modern Optics*, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[8] S. M. Rump, "Fast and parallel interval arithmetic", *BIT*, 1999, Vol. 39, No. 3, pp. 534-554.

[9] R. A. Serway, *Physics for Scientists and Engineers*, Saunders Publ., Philadelphia, 1996.

[10] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, Pasadena, California, 1997.

[11] A. R. Thompson, J. M. Moran, G. W. Swenson Jr., *Interferometry and Synthesis in Radio Astronomy*, Wiley, New York, 2001.

[12] G. Verschuur and K. I. Kellerman (eds.), *Galactic and extragalactic radio astronomy*, Springer Verlag, 1988.