Forward interval propagation through the discrete Fourier transform

Marco De Angelis
Institute for Risk and Uncertainty, University of Liverpool, United Kingdom.
marco.de-angelis@liverpool.ac.uk

Marco Behrendt
Institute for Risk and Reliability, Leibniz University Hannover, Germany.
behrendt@irz.uni-hannover.de

Liam Comerford
Institute for Risk and Uncertainty, University of Liverpool, United Kingdom.
l.comerford@liverpool.ac.uk

Yuanjin Zhang
School of Safety Science and Emergency Management, Wuhan University of Technology, China.
ylzhyj@126.com

Michael Beer
Institute for Risk and Reliability, Leibniz University Hannover, Germany.
beer@irz.uni-hannover.de

Abstract. In this paper an algorithm for the forward interval propagation on the amplitude of the discrete Fourier transform (DFT) is presented. The algorithm yields best-possible bounds on the amplitude of the DFT for real and complex valued sequences. We show that computing the exact bounds for the amplitude of the DFT can be achieved with an exhaustive examination of all possible corners of the interval-shaped domain. However, because the number of corners increases exponentially with the number of intervals, such method is infeasible for large interval signals. We provide an algorithm that does not need such an exhaustive search, and show that the best possible bounds for the amplitude can be obtained propagating complex pairs only from the convex hull of endpoints at each term of the Fourier series. Because the convex hull is always tightly inscribed in the respective rigorous bounding box resulting from interval arithmetic, we conclude that the obtained bounds are guaranteed to enclose the true values.

Keywords: Complex intervals; Dependency tracking; Convex hull; Interval algorithm.

1. Introduction

The analysis of engineering structures subject to harmonic stochastic excitation is often conveniently done in the Fourier domain (Newland(1993)). The response of the structure is obtained for each harmonic without the need for step-wise numerical integration. This kind of analysis however, seems to be incompatible with the fact that real data time-series records are often affected by

uncertainty. Some authors have successfully propagated interval uncertainty in dynamic structural analysis directly in the time domain (Muscolino et al.(2012)). In other instances, the literature on the topic seem to contemplate only purely probabilistic methods to address the problem of characterization of the uncertainty of the power spectrum from real data, as shown in (Comerford et al.(2015)Comerford, Kougioumtzoglou, and Beer) and literature therein. It is currently very difficult to convert an imprecise time signal to its Fourier domain. An algorithm for the interval Fourier transform is much needed to bridge the divide between imprecise signals and the well established harmonic structural analysis, and to enlarge the spectrum of engineering applications that explicitly account for uncertainty.

The discrete Fourier transform (DFT) is used in a variety of different applications in science and engineering, such as in spectral analysis, random vibration, differential equations, data compression, signal processing, image processing, probabilistic programming and many more (Sneddon(1995); James(2011); Newland(1993)). There are various algorithms available for transforming a signal with the DFT, of which the best known is probably the fast Fourier transfom (FFT), presented by Cooley & Tukey (Cooley and Tukey(1965)). Due to increasing computational power, simulations and equivalent calculations can be carried out ever faster, which is particularly important for the numerical analysis. An overview of a variety of algorithms used can be found in abundance in (Cormen et al.(2009)Cormen, Leiserson, Rivest, and Stein; Stoer and Bulirsch(2013)).

Real data records are required for a large number of problems in engineering. A major challenge with data acquisition that should not be neglected is that sensors only work accurately within certain tolerances and are therefore subject to uncertainties. These uncertainties arise, for instance, due to damaged sensors, device failures and measurement errors, or if the sensors are not precisely calibrated. In addition, the data could be captured incorrectly due to threshold limitations. For more accurate simulation results when utilising real data records, uncertainties must be taken into account (Comerford et al.(2015)Comerford, Kougioumtzoglou, and Beer).

In this work, three different methods are investigated to provide bounds on the amplitude of the DFT: (1) a brute-force method, (2) a method based on the convex hull, (3) a method based on complex interval arithmetic and rigorous bounding (Alefeld and Herzberger(2012); Moore(1966); Moore(1979); Moore et al.(2009)Moore, Kearfott, and Cloud). These methods provide a direct relationship between interval signals and the corresponding amplitudes of the DFT. Thus, intervals can replace the exact values when investigating structures while the signal's uncertainty can be captured by these intervals.

2. Problem statement

In this paper the focus is on interval-valued real signals denoted by$\overline{\underline{x}}_n$ (Alefeld and Herzberger(2012)). An interval-valued real number, or real interval, or simply an *interval* is a non-empty, closed, and bounded subset of the real line $\mathbb{R}$,

$$\overline{\underline{x}} := [\underline{x}, \overline{x}] := \{x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}\}.$$

Similarly a *complex interval* denoted by $\overline{\underline{z}}$, can be defined with a pair of intervals, one for the real component $\overline{\underline{z}}_{\mathrm{re}}$ and the other for the imaginary component $\overline{\underline{z}}_{\mathrm{im}}$,

$$\overline{z} := \overline{z}_{\text{re}} + i\ \overline{z}_{\text{im}} := \{z_{\text{re}} + i\ z_{\text{im}} \mid z_{\text{re}} \in \overline{z}_{\text{re}} \subseteq \mathbb{R}\ \&\ z_{\text{im}} \in \overline{z}_{\text{im}} \subseteq \mathbb{R}\},$$

for more about the space of complex intervals see (Alefeld and Herzberger(2012)). The interval extension of the DFT converts an interval signal (or ordered sequence) $\overline{x}_0, \overline{x}_1, ..., \overline{x}_n$ with $n < N$, into the respective signal of interval complex numbers $\overline{z}_0, \overline{z}_1, ..., \overline{z}_k$, with $k < N$, also known as Fourier sequence

$$\overline{z}_k := \sum_{n=0}^{N-1} \overline{x}_n\ e^{-\frac{i2\pi}{N}kn} = \sum_{n=0}^{N-1} \overline{x}_n\ \left(\cos\frac{2\pi}{N}kn - i\ \sin\frac{2\pi}{N}kn\right). \tag{1}$$

When x is a time signal, each complex number $z_k \in \mathbb{C}$ represents an harmonic of the signal with angular frequency $2\pi kn/N$. The DFT function is sometimes denoted by $\mathcal{F}$, while its interval extension by $\Diamond\mathcal{F}$, where $\overline{z}_k = \Diamond\mathcal{F}\,(\overline{x}_k)$.

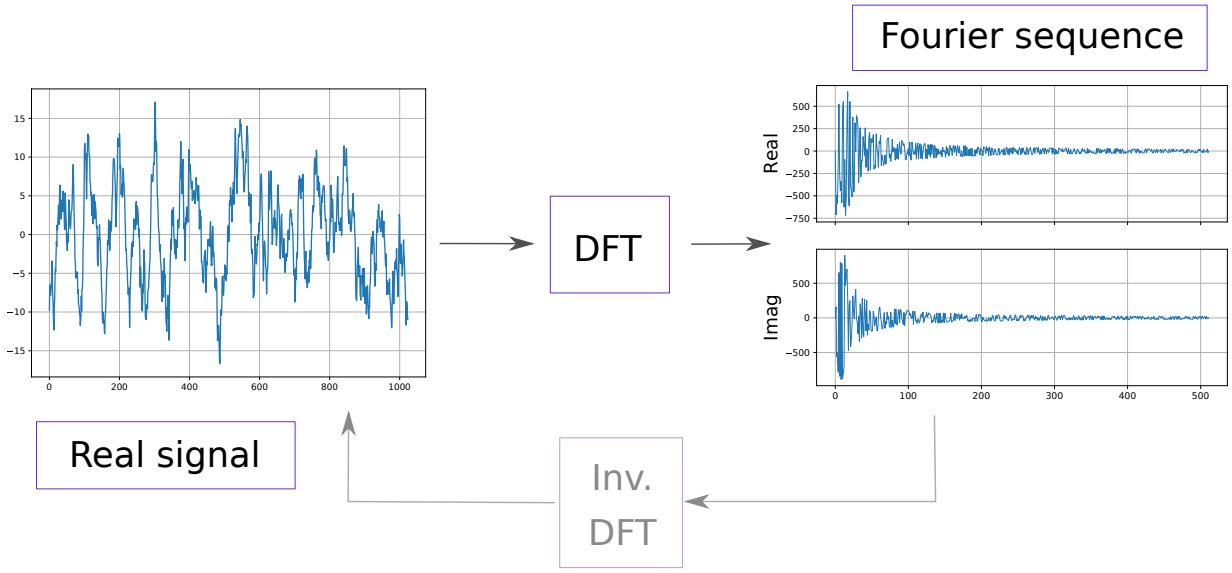

Figure 1. Discrete Fourier transform of a real signal to its Fourier sequence.

The DFT in practice is used to convert a time signal into its Fourier sequence (Figure 1) to study the harmonic properties of the signal where each element corresponds to a frequency. The DFT is often used to reduce the dimensionality of the signal; the inverse Fourier transform is used to reconstruct the signal after compression or simply to convert the signal into the time domain.

The objective of this work is to provide best-possible bounds on the interval extension for the *amplitude* of $\mathcal{F}$,

$$\text{abs}\,(\overline{z}_k) := |\overline{z}_k| := \sqrt{\left(\sum_{n=0}^{N-1} \overline{x}_n\ \cos\frac{2\pi}{N}kn\right)^2 + \left(\sum_{n=0}^{N-1} \overline{x}_n\ \sin\frac{2\pi}{N}kn\right)^2}. \tag{2}$$

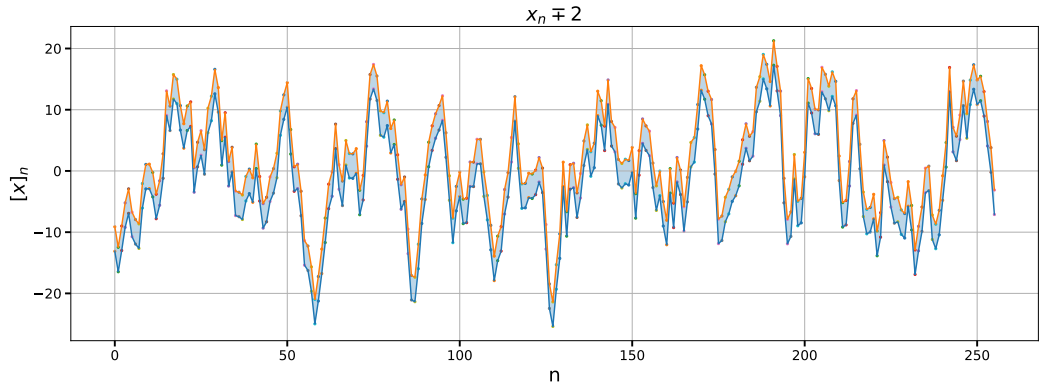

Figure 2. An interval signal $\overline{\underline{x}}_0, \overline{\underline{x}}_1, ..., \overline{\underline{x}}_n$, with N=128 points, and interval uncertainty ∓ 2.

The interval signal of Figure 2 will be considered throughout the manuscript. A convenient way to evaluate the interval extension $\Diamond\mathcal{F}$ is to split precise and uncertain components, as shown in Eq. (3). For example, when the signal is acquired using the same sensor (same precision) the interval uncertainty has the same value throughout. In such case the $\Diamond\mathcal{F}$ becomes

$$\overline{\underline{z}}_k := \sum_{n=0}^{N-1} x_n \; e^{-\frac{i2\pi}{N}kn} + \sum_{n=0}^{N-1} \xi \; \Delta \; e^{-\frac{i2\pi}{N}kn}, \tag{3}$$

where $\Delta = [-1,\ 1]$ is the unit interval, and $\xi \in \mathbb{R}$ is the precision of the interval signal, expressed in the same units as the signal. This form can be used to assess the effect of the cumulation of uncertainty on the amplitude as more and more uncertainty components get added.

3. Interval arithmetic

The interval extensions of Eqs.(1), (2), and (3) can be evaluated with interval arithmetic. The generalized rules of arithmetic for *addition*, *multiplication* and *power elevation* are needed for the purpose. The addition + between any two intervals $\overline{\underline{x}},\ \overline{\underline{y}}$ with $x,\ y \in \mathbb{R}$ is

$$\overline{\underline{x}} + \overline{\underline{y}} := [\underline{x} + \underline{y},\ \overline{x} + \overline{y}]. \tag{4}$$

The multiplication $*$ between an interval and a number $a \in \mathbb{R}$, which symbol is often omitted is

$$a * \overline{\underline{x}} := a\overline{\underline{x}} := \begin{cases} [a\underline{x},\ a\overline{x}],\ a > 0; \\ [a\overline{x},\ a\underline{x}],\ a < 0. \end{cases} \tag{5}$$

The power elevation with exponent two is

$$\overline{\underline{x}}^2 := \begin{cases} [\underline{x}^2,\ \overline{x}^2],\ \overline{\underline{x}} \geq 0 \\ [\overline{x}^2,\ \underline{x}^2],\ \overline{\underline{x}} < 0 \\ [0, \max\left(\underline{x}^2,\ \overline{x}^2\right),\ \overline{\underline{x}} \ni 0. \end{cases} \tag{6}$$

The square root of a positive real interval is

$$\sqrt{\underline{x}} := [\sqrt{\underline{x}},\ \sqrt{\overline{x}}],\ \underline{x} > 0. \tag{7}$$

With these rules the amplitude of the DFT interval extension can be evaluated using Eq. (2) without the need of a particular algorithm. The results provided by the interval arithmetic are guaranteed to enclose the exact bounds. However artefactual uncertainty due to the dependence problem makes the bounds obtained with interval arithmetic wider than they ought to be.

The interval extension of the DFT (Eq.(1)) can be computed using the complex addition operator. Consider two complex intervals $\underline{\overline{z}}_k = [\underline{z}_k, \overline{z}_k],\ k = 1, 2$, their addition is given by

$$\underline{\overline{z}}_1 + \underline{\overline{z}}_2 := [\underline{z}_1 + \underline{z}_2, \overline{z}_1 + \overline{z}_2].$$

With the complex addition operator the program used to compute $\mathcal{F}$ can be recycled to obtain its interval extension. The input *signal* to the program will be an ordered iterable of intervals.

```
1 def DFT(signal):  # Inputs an interval signal
2     F=[]
3     N = len(signal)  # length of the real signal
4     for k in range(N//2):  # for each frequency, with k=0,1,2,...
5         f = 0
6         for n in range(N):
7             f += signal[n]*exp(-1j*2*pi*k*n/N)
8         F.append(f)
9     return F
```

Listing 1: Code sample for a basic Python implementation of the Fourier transform.

Code sample List.1 can be used independently from external interval libraries, replacing an interval with an ordered list of a complex pair. The function in List.1 do need *numpy* for π and exp. The amplitude of $\mathcal{F}(\underline{\overline{\mathbf{x}}})$ for each frequency $\omega_k,\ k = 0, 1, 2, ...$ can be obtained replacing f at line 8 with abs(f), or with the following function.

```
1 def DFT_amplitude(signal):
2     F = DFT(signal)
3     return [abs(f) for f in F]
```

Listing 2: Code sample for the DFT amplitude.

The only caveat in List.1 program, is that the length of the signal must be a power of two in order for the signal to be fully transformed to the Fourier domain. Slightly modified programs can be obtained with little extra effort to map signals that are not divisible by two, see e.g. (Newland(1993)).

3.1. Notes on the dependence problem using interval arithmetic

The bounds on the amplitude of the interval extension $\mathcal{F}$ evaluated with interval arithmetic are *rigorous* in the sense that all the possible signals $\mathbf{x} \in \underline{\overline{\mathbf{x}}}$ are mapped inside these bounds. However when the amplitude is computed, the bounds obtained with Eq. (2) will be suboptimal because of

the dependence problem. The computation of Eq. (2) with interval arithmetic does not track the dependence between the real and the imaginary component, thus the bounds are outer-estimated. The bounds are rigorous because all of the possible dependencies between real and imaginary components are considered by the interval arithmetic bounds.

4. The brute-force method

The *brute-force* method is too inefficient to be used in practical applications, and it is herein used mainly for illustration purposes. The method tracks the dependency between real and imaginary components by constructing a binary tree of all the propagated endpoints at each iteration step. Clearly the number of endpoints increases exponentially in base two and with exponent given by the iteration number. The total number of iterations is the length of the signal, so the longer the signal the more inefficient is the *brute-force* method. Thus this method can only be used for research purposes either on a very short signal or on a signal with very few intervals. On a personal computer the recommended size of the problem is eight, more iterations may slow down the process significantly.

```
def BruteForce(intervalsignal, k, Limit):
    N = len(intervalsignal)  # <- length of the signal
    COMB=[]
    pair = [complex(exp(-2*pi*1j*k*0/N)) * iend for iend in intervalsignal[0]]
    pairs = pair  # initialise set of endpoints
    COMB.append(pairs)
    for n in range(1,Limit): # Limit < N when N is large
        pair  =  [complex(exp(-2*pi*1j*k*n/N)) * iend for iend in intervalsignal[n
    ]]
        pairs = [[pair[0] + ps for ps in pairs],[pair[1] + ps for ps in pairs]]
        COMB.append(pairs) # add to list of endpoints
        pairs = pairs[0] + pairs[1] # update for next iteration
    return COMB
```

Listing 3: Code sample for Python implementation of the *brute-force* method

Because the *brute-force* method is so inefficient, we do not provide the code for computing the bounds out of the set of endpoints output by the program. The code List.3 can be used without external interval libraries; however it does require *numpy* for π and exp. The program 3 outputs the sets of points depicted in Figure 3.

Figure 3 shows that the bounds provided by interval arithmetic in the complex plane are rigorous and best possible. The Figure superimposes the propagated endpoints through the Fourier transform on the interval arithmetic box, for a given *frequency* ($k = 9$) of the Fourier domain, and up to eight iterations. This is because there are no repeated variables in the complex expression of Eq. (1). Because the DFT is a linear function, mapping an *interval* signal to its Fourier counterpart requires the information from the endpoints only. The number of interval components present in the signal does not seem to affect the inflation of the bounds that keep enclosing the endpoints tightly. The final interval box, for a given frequency, seems to always tightly enclose the set of propagated

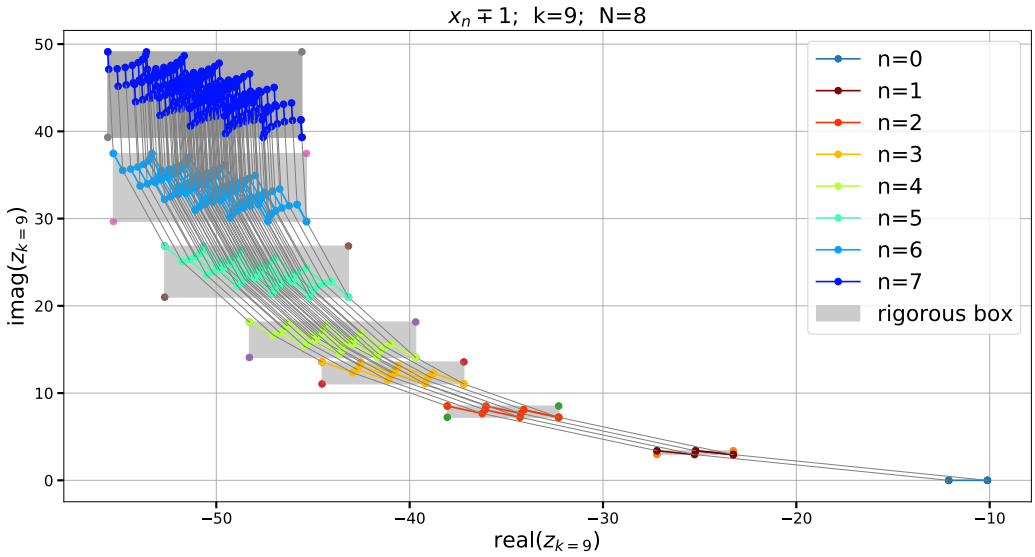

Figure 3. Set of endpoints produced by the *brute-force* method v. rigorous box.

endpoints. For example if the signal had eight components, given $k = 9$, we would be looking at the last gray box ($n = 7$).

5. The selective method

We observe that only the endpoints on the convex hull on the current set of endpoints carry the information about the bounds. Numerical experiments performed using the *brute-force* method have confirmed this intuition, however the investigation is limited by the number of iterations allowed by the brute force. Thankfully, interval arithmetic comes to the rescue. In fact, we can keep propagating the endpoints on the convex hull as well as the interval arithmetic box for as many iterations as needed and for however large signals. Moreover, because interval arithmetic is the most efficient method its box can always be computed alongside the *selective* method.

The numerical experiments done running the *selective* method alongside interval arithmetic have confirmed and *verified* the intuition that the selective method yields rigorous best possible bounds in each tested case. Figure 4, shows on four different frequencies, the three methods altogether. The convex hull is the set of points resting on the set perimeter, which in Figure 4 is depicted by a red closed line. The convex hull is obtained at each iteration by propagating only the endpoints on the perimeter of the previous iteration step.

Interval arithmetic gives us the verified guarantee that the *selective* method is rigorous, i.e. does not underestimate the uncertainty. The fact that the *selective* method makes use of a subset of the endpoints ensures that its bounds are also best possible. In fact, it is always possible to index the

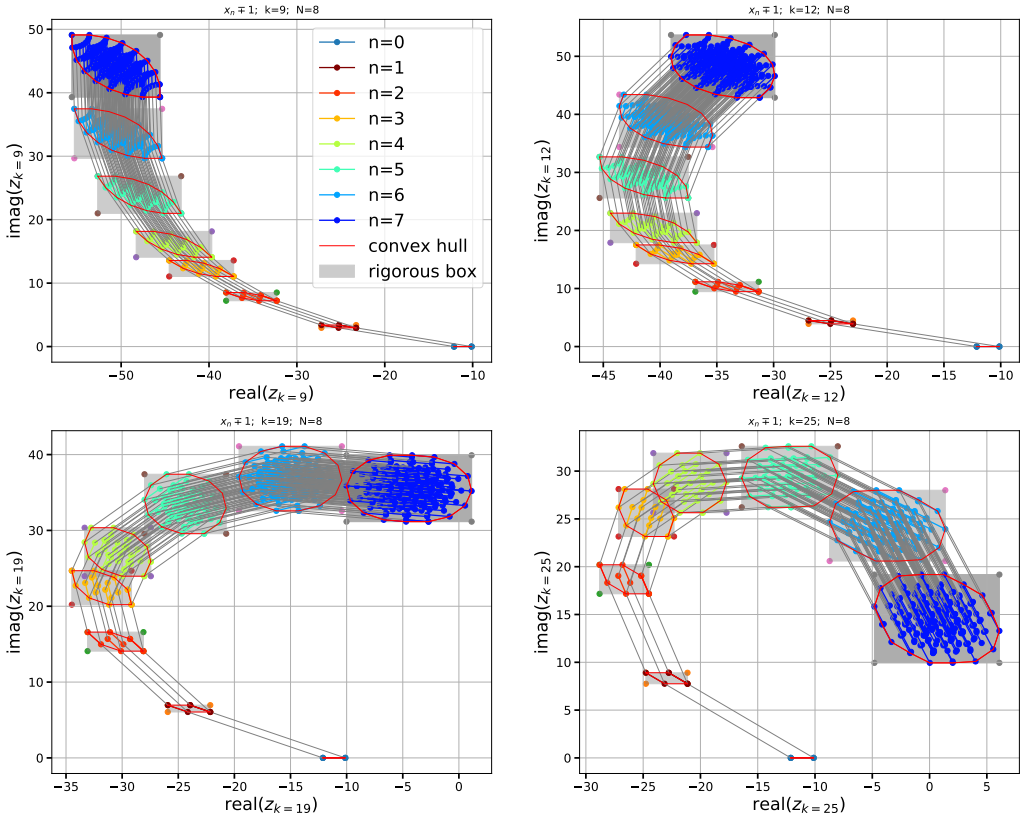

Figure 4. Set of points on the convex hull and interval box for four different frequencies.

convex-hull endpoints resting on the edge of the rigorous box and propagate these points back to identify the combination of endpoints yielding the bounds. However because there is no guarantee that theses endpoints will be unique, the algorithm cannot be used directly to perform the inverse transform.

The code produces the set of points in the convex hull as shown in List.4 and can be obtained as a derivation from the *brute-force* method. The function *Selective* in the code snippet List.4 will be invoked for each frequency k of the Fourier sequence.

```
from scipy.spatial import ConvexHull
def Selective(intervalsignal, k):
    N = len(intervalsignal)
    CHULL = []
    pair = [complex(exp(-2*pi*1j*k*0/N)) * iend for iend in intervalsignal[0]]
```

```
    chullpair = pair
    CHULL.append(chullpair)
    for n in range(1,N):
        pair =  [complex(exp(-2*pi*1j*k*0/N)) * iend for iend in intervalsignal[n
    ]]
        chullpair=[[pair[0]+ch for ch in chullpair],[pair[1]+ch for ch in
    chullpair]]
        chullpair = chullpair[0] + chullpair[1]
        chullpairs_RI = [[p.real, p.imag] for p in chullpair]
        hull = ConvexHull(chullpairs_RI)
        chullpair = [chullpairs_RI[h][0] + 1j*chullpairs_RI[h][1] for h in hull.
    vertices]
        CHULL.append(chullpair)
    return CHULL
```

Listing 4: Code snippet for the Python implementation of the *selective* method.

6. Rigorous bounding method

There are no repeated variables in the interval extension of Eq. (1), thus the final enclosing box is the smallest box possible containing the set of points generated by the *brute force* method. In List.5 we show the procedure to obtain the enclosing interval box at each iteration of the Fourier transform. Note that if the enclosing box was not the smallest or *best possible*, there would have been a gap between the box and the convex hull of points, effectively making it impossible for us to verify the results produced by the selective method. Luckily in our numerical experiments there has been no instance of such a case. The rigorous bounding method can be performed alongside the selective method with very little additional computations, to provide a prompt verification certificate of the bounds outputted by the selective method.

```
def RigorousBox(intervalsignal, k):
    N = len(intervalsignal)
    RBOX = []
    firstinterval = [complex(exp(-2*pi*1j*k*0/N)) * iend for iend in
    intervalsignal[0]]
    RBOX.append(firstinterval)
    ci =firstinterval
    for n in range(1,N):
        ci += complex(exp(-2*pi*1j*k*n/N)) * intervalsignal[k]
        RBOX.append(ci)
    return RBOX
```

Listing 5: Code snippet for the Python implementation of the *interval* method.

The amplitude of the Fourier sequence can be obtained just using the information carried by the corners of the enclosing box. Because the set of extreme endpoints rest on the convex hull, the amplitude obtained with the enclosing box will carry inflated uncertainty. The bounds on the amplitude will be reflective of the fact that the zone outside the convex hull between the box will be propagated too, effectively making the bounds on the amplitude much puffier. The extra puffiness

produced by this method however is compensated by the efficiency of this method, which practically carries no additional computational cost when compared to the *precise* DFT.

7. The final algorithm for the bounds on the amplitude

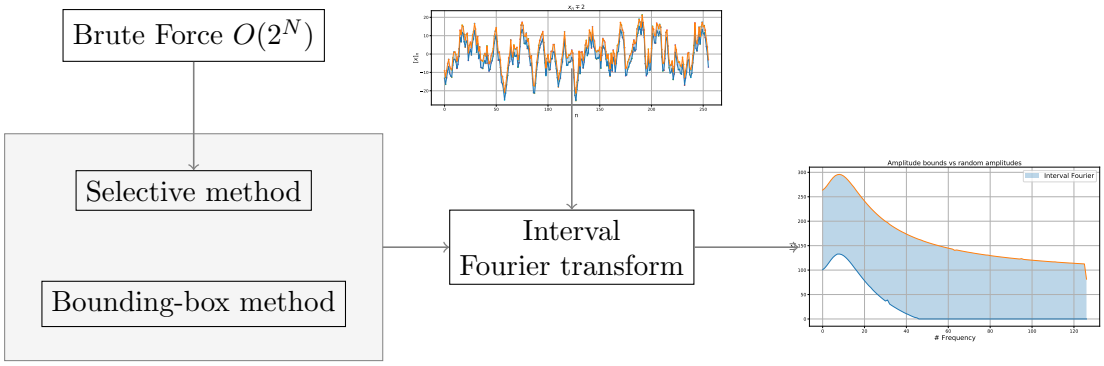

Figure 5. Interval Fourier transform algorithm in summary.

In this section we show how to obtain the bounds on the amplitude for each frequency. A schematic representation of how the three methods interact with each other is provided in Figure5.

The convex hull or the interval box at the last iteration for $n = N - 1$, is used to compute the amplitude at each frequency. We have shown that the endpoints on the convex hull yield the best possible bounds, thus the bounds on the amplitude are obtained by taking the *minimum* and *maximum* amplitude corresponding to this set of endpoints. The points corresponding to the *extrema* are shown in Figure6. Because the amplitude of the Fourier transform is effectively the Euclidean norm on the $\mathbb{R}^2$ isomorphism of the complex plane, *minimum* and *maximum* correspond to the points that are nearest and furthest from the origin of the complex plane. However, this is true as long as the convex hull does not enclose the origin. When the convex hull encloses the origin the *minimum* will be attained on the nearest endpoint to the origin. Because the *selective* approach "forgets" about all the endpoints in the interior of the convex hull, it is not possible to determine exactly the nearest to the origin endpoint.

The algorithm for the amplitude bounds will therefore assume that the *minimum* is attained at the origin of the complex plane in such situation, as shown in Figure7.

A simple implementation of the algorithm for the amplitude bounds with the selective method is shown in List.6. The function *AmplitudeBounds* needs an external function called *origin_in_complex_hull* that returns *True* when the convex hull encloses the origin and a *False* otherwise.

```
def AmplitudeBounds(intervalsignal, k):
    CHULL = Selective(intervalsignal, k)
    convhull = CHULL[-1]
    chull_max = numpy.argmax([abs(h) for h in convhull])
    chull_min = numpy.argmin([abs(h) for h in convhull])
    if origin_in_complex_hull(convhull):
        return 0, abs(convhull[chull_max])
```

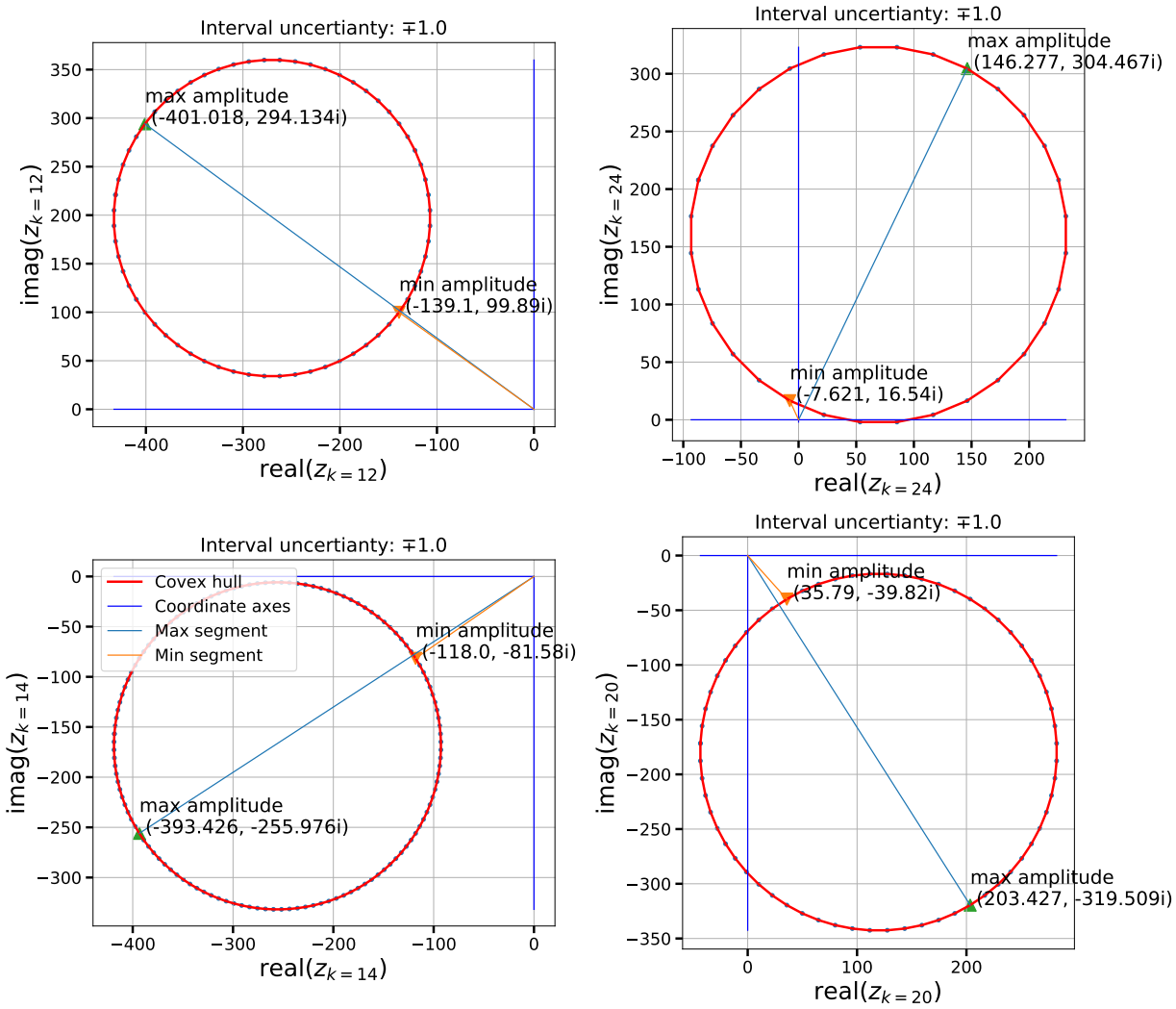

Figure 6. Convex hull of endpoints for an interval signal N=128 long, and four different frequencies.

```
    else:
        return abs(convhull[chull_min]), abs(convhull[chull_max])
```

Listing 6: Python function for the amplitude bounds with the *selective* method.

The procedure for the amplitude bounds with the interval box is equivalent except that is much easier to check if the origin is contained in the box. Iterating over all the frequencies of the Fourier sequence as shown in List.7 leads to the amplitude bounds shown in Figure8.

```
def DFT_bounds(intervalsignal):
    N = len(intervalsignal)
    BOUNDS_I, BOUNDS_C = [],[]
```

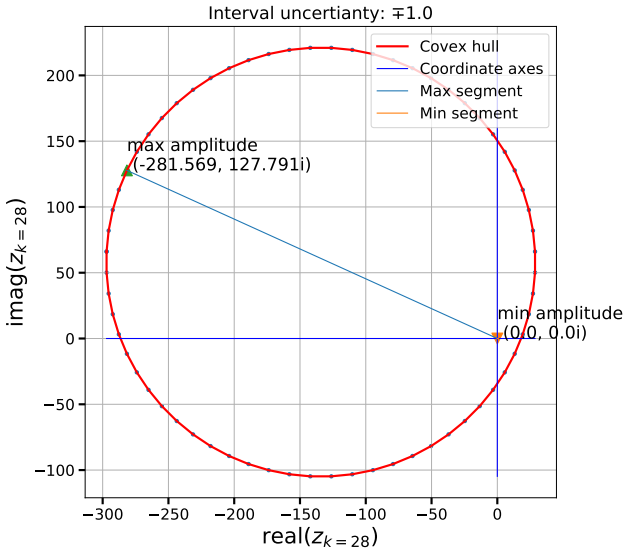

Figure 7. Convex hull of endpoints enclosing the origin of the complex plain.

```
    for k in range(1,N//2):
        bI,bC = AmplitudeBounds(intervalsignal, k)
        BOUNDS_I.append(bI)
        BOUNDS_C.append(bC)
    return BOUNDS_I, BOUNDS_C
```

Listing 7: Python function for the amplitude bounds for every frequency.

8. Conclusions

In this paper we have presented three methods for computing the bounds of the Fourier amplitude when an interval signal is provided. We have shown how the problem can be tackled with a brute-force method. However the brute-force algorithm is NP-complete, thus only working for very small signals (< 20 in length), with no applications in practice. Despite its inefficiency the brute-force method has provided the inspiration for the second more efficient *selective* method, which yields the best possible bounds on the amplitude of the Fourier sequence. In this paper we have questioned the rigour or reliability of the *selective* algorithm, which has led us to our third method, which uses complex interval arithmetic to propagate the bounds through the Fourier transform. The third method called *bounding box*, has provided a viable efficient tool to verify the rigorousness of the *selective* method even for large interval signals. The *bounding box* method has also provided a robust alternative to obtain the amplitude bounds when efficiency is a priority at the expense of losing specificity in the bounds of the amplitude.

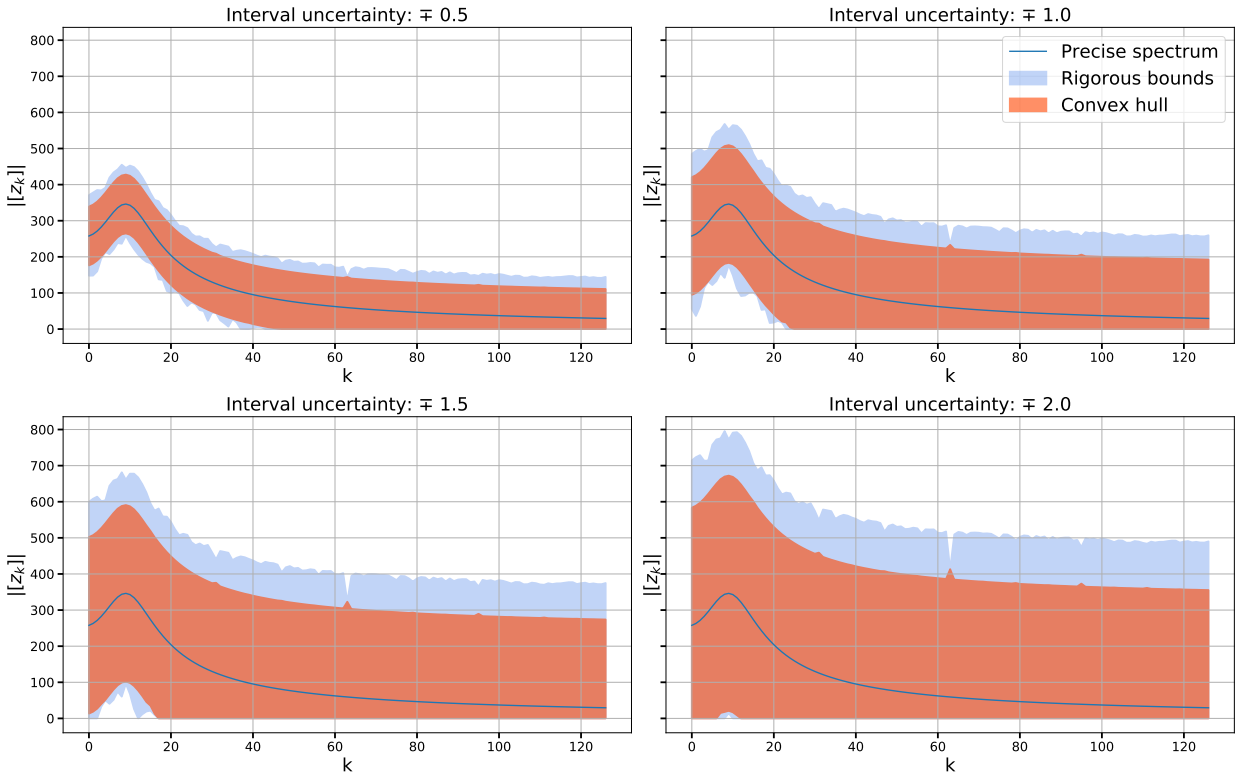

Figure 8. Amplitude bounds for the Fourier transform for different uncertainty levels.

9. Acknowledgements

This research is funded by the Engineering & Physical Sciences Research Council (EPSRC) with grant no. EP/R006768/1. The research is also funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) with grant no. BE 2570/4-1 and CO 1849/1-1. EPSRC and DFG are greatly acknowledged for their funding and support.

References

D. Newland, *An Introduction to Random Vibrations, Spectral and Wavelet Analysis*, Dover books on engineering (Longman Scientific & Technical, 1993), ISBN 9780582215849.

G. Muscolino, R. Santoro, and A. Sofi, in *5th International conference on Reliable Engineering Computing (REC2012). Edited by Brno, Czech Republic: Ing. Vladislav Pokorný-LITERA* (2012), 407–25.

L. Comerford, I. A. Kougioumtzoglou, and M. Beer, On quantifying the uncertainty of stochastic process power spectrum estimates subject to missing data, *International Journal of Sustainable Materials and Structural Systems* **2**, 185 (2015).

I. N. Sneddon, *Fourier transforms* (Courier Corporation, 1995).

J. F. James, *A Student's Guide to Fourier Transforms: With Applications in Physics and Engineering*, Student's Guides (Cambridge University Press, 2011), 3rd ed.

J. W. Cooley and J. W. Tukey, An algorithm for the machine calculation of complex fourier series, *Mathematics of computation* **19**, 297 (1965).

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms* (MIT press, 2009).

J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, vol. 12 (Springer Science & Business Media, 2013).

G. Alefeld and J. Herzberger, *Introduction to Interval Computation*, Computer Science and Applied Mathematics (Elsevier Science, 2012), ISBN 9780080916361.

R. E. Moore, *Interval analysis*, vol. 4 (Prentice-Hall Englewood Cliffs, 1966).

R. E. Moore, *Methods and applications of interval analysis* (SIAM, 1979).

R. Moore, R. Kearfott, and M. Cloud, *Introduction to Interval Analysis*, Other titles in applied mathematics (Cambridge University Press, 2009), ISBN 9780898716696.