Fourier Transform and Other Quadratic Problems under Interval Uncertainty

Presenter: Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
vladik@utep.edu
https://www.cs.utep.edu/vladik

(based on joint work with
Oscar Galindo and Christropher Ibarra from our department
and Michael Beer from Leibniz University of Hannover, Germany)

1. What Is Fourier Transform: Reminder

- Newton showed that every light can be represented as a combination of pure colors (history: rainbow in the Bible).
- He passed the usual light through a prism and got a rainbow.
- Then, he used another prism to combine the colored rays and got back the original white light.
- In mathematical terms, pure color corresponds to a sinusoid $x(t) = A \cdot \sin(\omega \cdot t + \varphi)$, i.e., equivalently, to

$$x(t) = a \cdot \cos(\omega \cdot t) + b \cdot \sin(\omega \cdot t).$$

- Fourier showed that, indeed, every reasonable function can be represented as a linear combination of sinusoids:

$$x(t) = \sum_{\omega} (a(\omega) \cdot \cos(\omega \cdot t) + b(\omega) \cdot \sin(\omega \cdot t)).$$

2. What Is Fourier Transform (cont-d)

- He also showed how to compute the coefficients $a(\omega)$ and $b(\omega)$ based on the values $x_i = x(t_0 + i \cdot \Delta t)$, $i = 1, \ldots, n$:

$$a_m \stackrel{\text{def}}{=} a(m \cdot \Delta\omega) = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \cdot \cos\left(\frac{2\pi}{n} \cdot i \cdot m\right);$$

$$b_m \stackrel{\text{def}}{=} b(m \cdot \Delta\omega) = -\frac{1}{n} \cdot \sum_{i=1}^{n} x_i \cdot \sin\left(\frac{2\pi}{n} \cdot i \cdot m\right).$$

- These formulas are called *Fourier transform.*
- Based on the values a_m and b_m, we can form $A_m = \sqrt{a_m^2 + b_m^2}$.
- The original formulas require n^2 steps, too many for large n.
- In the 1960s, Fast Fourier Transform (FFT) algorithm was invented that takes time $O(n \cdot \log_2(n)) \ll n^2$.
- FFT is one of the main data processing techniques in science and engineering.

3. Need to Take Interval Uncertainty into Account

- The values $x_i = x(t_0 + i \cdot \Delta t)$ come from measurements.
- Measurements are never 100% accurate.
- The measurement results $\widetilde{x}_i$ are, in general, different from the actual (unknown) values x_i.
- In other words, there is a non-zero measurement error $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$.
- Often, the only information that we have about Δx_i is the upper bound: $|\Delta x_i| \le \Delta_i$.
- In this case, after the measurement, the only information we gain about the actual value x_i is that $x_i \in [\underline{x}_i, \overline{x}_i] = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.
- For different values x_i from these intervals, we get, in general, different values of a_m, b_m, and A_m.
- A natural question is: what are the ranges $[\underline{a}_m, \overline{a}_m]$, $[\underline{b}_m, \overline{b}_m]$, and $[\underline{A}_m, \overline{A}_m]$ of possible values of these quantities?

4. Computations Are Only Approximate

- Of course, computers only represent rational numbers, which the values of sine and cosine are usually irrational.
- Thus, we can only compute these ranges with a given accuracy $\varepsilon > 0$.
- For a_m, b_m, and A_m, feasible algorithms are known.
- In this talk, we show how these algorithms can be extended to a more general case.

5. Some of These Questions Are Easy

- The coefficients a_m and b_m linearly depend on x_i.
- A general linear function has the form:

$$y = c_0 + \sum_{i=1}^{n} c_i \cdot x_i.$$

- When we plug in the measurement results, we get

$$\widetilde{y} = c_0 + \sum_{i=1}^{n} c_i \cdot \widetilde{x}_i.$$

- For $x_i = \widetilde{x}_i - \Delta x_i$, we get

$$y = c_0 + \sum_{i=1}^{n} c_i \cdot \widetilde{x}_i - \sum_{i=1}^{n} c_i \cdot \Delta x_i = \widetilde{y} - \sum_{i=1}^{n} c_i \cdot \Delta x_i.$$

- Here, $\Delta x_i \in [-\Delta_i, \Delta_i]$.

6. Some of These Questions Are Easy (cont-d)

- The value of the sum is the largest when each term $c_i \cdot \Delta x_i$ is the largest.
- When $c_i > 0$, the term is increasing, so maximum is attained for $\Delta x_i = \Delta_i$ and is equal to $c_i \cdot \Delta_i$.
- When $c_i < 0$, the term is decreasing, so maximum is attained for $\Delta x_i = -\Delta_i$ and is equal to $-c_i \cdot \Delta_i$.
- In both cases, we have $|c_i| \cdot \Delta_i$.
- Thus, the smallest possible value of y is $\underline{y} = \widetilde{y} - \Delta$, where

$$\Delta \stackrel{\text{def}}{=} \sum_{i=1}^{n} |c_i| \cdot \Delta_i.$$

- In general, the range of possible values for y is $[\underline{y}, \overline{y}] = [\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.
- This is computable in linear time.

7. Some of These Questions Are Not So Easy

- Maximizing A_m is equivalent to maximizing its square A_m^2.
- The problem is that A_m^2 is a quadratic function of x_i's.
- For quadratic functions, in general, computing the range under interval uncertainty is NP-hard.
- It is NP-hard even for computing the range of sample variance:

$$V = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 .$$

- We show that for A_m^2, computing the range is feasible.
- To show this, we will describe a class of quadratic expressions – containing computing A_m^2 – for which range can be feasibly computed.

8. Class of Quadratic Expressions for Which the Range Can Be Feasibly Computed

- A general quadratic function has the form

$$f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} \cdot x_i \cdot x_j + \sum_{i=1}^{n} c_i \cdot x_i + c_0.$$

- The expression for A_m^2 is the sum of two squares of linear expressions: $A_m^2 = a_m^2 + b_m^2$.
- This implies that the rank of the corresponding matrix $c_{i,j}$ is 2 – i.e., we only have two non-zero eigenvalues.
- The general class is when the matrix $c_{i,j}$ has rank k, i.e., that it has k non-zero eigenvalues λ_j, $j = 1, \ldots, k$.
- We will denote the corresponding unit eigenvectors by $(e_{j,1}, \ldots, e_{j,n})$.

9. What We Do in This Talk

- We will show that for any fixed k, there is a feasible algorithm for estimating the range of the corresponding quadratic expression.
- This algorithm takes time $O(n^k)$ in the homogeneous case and $O(n^{k+1})$ in the general case.
- So, as k increases, the time grows fast, and for $k \approx n$, we get exponential time.
- This makes sense: since the problem is NP-hard, we cannot expect lower-than-exponential computation time.

10. Facts from Calculus: Reminder

- Computing the minimum of f is equivalent to computing the maximum of $-f$.
- Thus, it is sufficient to be able to compute the maximum.
- According to calculus, the maximum with respect to each variable $x_i \in [\underline{x}_i, \overline{x}_i]$ is attained:
  - either for $x_i = \underline{x}_i$, then $\dfrac{\partial f}{\partial x_i} \le 0$;
  - or for $x_i = \overline{x}_i$, then $\dfrac{\partial f}{\partial x_i} \ge 0$;
  - or for $x_i \in (\underline{x}_i, \overline{x}_i)$, then $\dfrac{\partial f}{\partial x_i} = 0$.

11. Let Us Apply These Facts to Our Problem

- We start with the quadratic expression

$$f=\sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}\cdot x_i\cdot x_j+\sum_{i=1}^{n} c_i\cdot x_i+c_0.$$

- In terms of eigenvalues and eigenvectors, the quadratic expression takes the form

$$f=\sum_{j=1}^{k}\lambda_j\cdot\left(\sum_{i=1}^{n} e_{j,i}\cdot x_i\right)^2+\sum_{i=1}^{n} c_i\cdot x_i+c_0.$$

- Its partial derivative w.r.t. x_i is equal to:

$$\frac{\partial f}{\partial x_i}=2\sum_{j=1}^{k}\lambda_j\cdot\left(\sum_{\ell=1}^{n} e_{j,\ell}\cdot x_\ell\right)\cdot e_{j,i}+c_i.$$

- This expression can be described in terms of n $(k+1)$-dimensional vectors

$$e_i=(e_{1,i},\ldots,e_{k,i},c_i)\text{ and }e_i^*=(2\lambda_1\cdot e_{1,i},\ldots,2\lambda_k\cdot e_{k,i},0).$$

12. Let Us Apply These Facts to Our Problem (cont-d)

- In terms of the dot (scalar) product, we get $\dfrac{\partial f}{\partial x_i} = e_i \cdot S$, where:

$$S \stackrel{\text{def}}{=} \sum_{\ell=1}^{n} x_\ell \cdot e^*_\ell + (0, \ldots, 0, 1).$$

- Thus, all the $(k+1)$-dimensional points e_i for which $\dfrac{\partial f}{\partial x_i} = 0$ are located on a k-dimensional plane $\{e : e \cdot S = 0\}$.
- Let us first consider the non-degenerate case, when every group of $k+1$ vectors e_i is linearly independent.
- We can have no more than k linearly independent vectors on the same k-dimensional plane.
- Thus, we can have no more than k indices i for which partial derivative is 0.

13. Let Us Apply These Facts to Our Problem (cont-d)

- For points on one side of the plane, where $\dfrac{\partial f}{\partial x_i} < 0$, maximum is attained for $x_i = \underline{x}_i$.
- For points on the other side of the plane, where $\dfrac{\partial f}{\partial x_i} > 0$, maximum is attained for $x_i = \overline{x}_i$.
- If there are fewer than k points at which the derivative is 0, we can move the plane a little bit until it reaches exactly k points.
- So, we arrive at the following algorithm.

14. Resulting Algorithm: Non-Degenerate Case

- *Given:*
  - a quadratic expression with matrix of rank k:
$$f = \sum_{i=1}^n \sum_{j=1}^n c_{i,j} \cdot x_i \cdot x_j + \sum_{i=1}^n c_i \cdot x_i + c_0; \text{ and}$$
  - intervals $[\underline{x}_i, \overline{x}_i]$.
- *Find:* the range $[\underline{y}, \overline{y}]$ of the expression f.
- We consider all possible selections $1 \le i_1 < \ldots < i_j < \ldots < i_k \le n$ of k different indices.
- There are $O(n^k)$ such selections.
- For each selection, we solve a system of k linear equations with k unknowns $S_1, \ldots, S_k$:
$$\sum_{j'=1}^k e_{j',i_j} \cdot S_{j'} + c_{i_j} = 0, \quad j = 1, \ldots, k.$$

15. Algorithm, Non-Degenerate Case (cont-d)

- We consider all 3^k possible divisions of the set $\{1,\ldots,k\}$ into 3 subsets L (lower), U (upper), and I (inside).
- For each division, we consider two possible signs $\varepsilon \in \{-,+\}$.
- For each division and sign:
  - we set $x_i = \underline{x}_i$ if $(e_i \cdot S < 0$ and $\varepsilon = +)$ or $(e_i \cdot S > 0$ and $\varepsilon = -)$;
  - we set $x_i = \overline{x}_i$ if $(e_i \cdot S > 0$ and $\varepsilon = +)$ or $(e_i \cdot S < 0$ and $\varepsilon = -)$;
  - we set $x_{i_j} = \underline{x}_{i_j}$ for $j \in L$ and $x_{i_j} = \overline{x}_{i_j}$ for $j \in U$;
  - the remaining values x_{i_j} for $j \in I$, from the system of equations:

$$\frac{\partial f}{\partial x_{i_j}} = 2\sum_{j'=1}^{k} \lambda_{j'} \cdot \left(\sum_{\ell=1}^{n} e_{j',\ell} \cdot x_\ell\right) \cdot e_{j',i_j} + c_{i_j} = 0, \quad j = 1,\ldots,k;$$

  - if the resulting values x_{i_j} are in $[\underline{x}_{i_j}, \overline{x}_{i_j}]$, then we compute the value $f(x_1,\ldots,x_n)$.

16. Algorithm, Non-Degenerate Case (cont-d)

- The largest of the corresponding values of the expression f is $\overline{y}$, the smallest is $\underline{y}$.
- Computing f by using eigenvectors takes time $O(n \cdot k) = O(n)$.
- We perform it for all $O(n^k) \cdot 2 \cdot 3^k = O(n^k)$ cases, so overall time is $O(n^{k+1})$, which is feasible.

17. General Case

- For each $\delta > 0$, we can add δ-small random changes to the values c_{ij} and c_i.
- For example, we can add values uniformly distributed on the interval $[-\delta, \delta]$.
- With probability 1, the resulting system is non-degenerate.
- The difference between the original and new objective functions does not exceed
$$\delta \cdot \left(\sum_{i=1}^{n} \sum_{j=1}^{n} |x_i| \cdot |x_j| + \sum_{i=1}^{n} |x_i| \right).$$
- We can use straightforward interval computations to get the bound B on the expression in parentheses.
- So, for any given $\varepsilon > 0$, if we take $\delta = \varepsilon / B$, we get a non-degenerate objective function which is ε-close to the original one.

18. General Case (cont-d)

- The bounds for the new objective function are ε-close to the bounds on the original one.
- Thus, we have a feasible $O(n^{k+1})$ algorithm for computing $\underline{y}$ and $\overline{y}$ with any given accuracy $\varepsilon > 0$.

19. Homogeneous Case

- In the Fourier transform case, $c_i = c_0 = 0$, so $f = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{i,j} \cdot x_i \cdot x_j$.
- In such *homogeneous* case, we can consider k-dimensional vectors

$$e_i = (e_{1,i}, \ldots, e_{k,i}) \text{ and } e_i^* = (2\lambda_1 \cdot e_{1,i}, \ldots, 2\lambda_k \cdot e_{k,i}).$$

- In non-degenerate case, we thus have $\le k-1$ indices i at which the derivative is 0.
- So, we have a similar algorithm, but with $k-1$ instead of k.
- This algorithm requires time $O(n^k)$.

20. Back to Fourier Transform

- For Fourier transform, we get the sum-of-squares expression with

$$e_i = \left(\cos\left(\frac{2\pi}{n}\cdot i\cdot m\right), \sin\left(\frac{2\pi}{n}\cdot i\cdot m\right)\right).$$

- Different vectors e_i are non-degenerate.
- Some of these vectors coincide; they are multiplied by the sum X_a of the corresponding value x_i, $i \in S_a$.
- For these sums, the range $[\underline{X}_a, \overline{X}_a]$ is the sum of the ranges $[\underline{x}_i, \overline{x}_i]$:

$$\underline{A}_a = \sum_{i\in S_a} \underline{x}_i; \quad \overline{X}_a = \sum_{i\in S_a} \overline{x}_i.$$

- So, for Fourier transform under interval uncertainty, we get an $O(n^2)$ algorithm. (Actually, it can be reduced to linear time.)