International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Memory Complexity and Precomputation in Modern FFT Architectures

Tajinder Singh

Assistant Professor
Department of Mathematics
Government college, Hoshiarpur

Abstract:
This review reconsiders how fast Fourier transform (FFT) algorithms ought to be evaluated when the target hardware is no longer dominated by the cost of a complex multiplier. The classical literature, surveyed canonically by Duhamel and Vetterli[1], framed progress in terms of arithmetic reduction: multiplication counts first, then additions. We argue that this metric has lost much of its diagnostic value for the algorithms that now drive non-uniform sampling, sparse spectral recovery, and three-dimensional grid convolution. The relevant cost has migrated into the memory footprint of precomputed window tables, into the structure of sparse selection matrices, and into the oversampled grids that absorb interpolation error. We synthesise results from non-equispaced FFT theory, sparse deterministic schemes, biomolecular grid integration, and interferometric visibility simulation to show that modern speed-ups tend to be paid for in storage depth and precomputation overhead rather than in butterfly arithmetic.

Keywords: Fast Fourier transform, memory complexity, precomputation overhead, arithmetic complexity, non-uniform FFT.

1 Introduction

The fast Fourier transform has been a settled subject, at least in its one-dimensional form, for several decades. Duhamel and Vetterli's 1990 tutorial[1] closed off a long arc of optimisation that began with Cooley and Tukey's 1965 reduction from $O(N^2)$ to $O(N\log_2 N)$ for composite-length DFTs and ended, in practical terms, with the split-radix algorithm and the Winograd lower bounds on multiplicative complexity. The unifying performance metric throughout this period was arithmetic. How many real multiplications, and secondarily how many real additions, were required to evaluate a length-N DFT? That was the question, and almost every contribution from Bergland's higher-radix variants through Yavne's 1968 record for length-2^n transforms answered it incrementally.

The reasons for this preoccupation were technological. Multiplication on the hardware of the 1960s and 1970s was substantially more expensive than addition, and silicon area for a complex multiplier dwarfed that of an adder[1]. The split-radix construction[1] set a multiplication record for length-2^n transforms that, as the authors noted, no subsequent practical algorithm has improved upon. Winograd's later work supplied tight lower bounds on the multiplicative side, letting the community measure how far a given algorithm sat from optimality. By 1990, the picture looked closed.

Several observations made within that same review hinted, however, at the metric's encroaching obsolescence. Duhamel and Vetterli themselves remarked that addition counts and memory access patterns were already as important as multiplication counts in software, and that communication cost dominated in VLSI implementations[1]. A length-1024×1024 two-dimensional DFT, they observed in passing, already required on the order of 10^6 storage words; at that scale, the access pattern rather than the multiplier count governed wall-clock time. The Winograd Fourier Transform Algorithm, which minimised multiplications, was judged impractical in part because of its huge memory demand and inflated addition count. These were not peripheral concerns but early symptoms of a shift in the relevant resource.

Moving past the radix-2 boundary, the algorithms that now dominate non-trivial signal-processing pipelines no longer admit a meaningful multiplier-only accounting. The non-equispaced FFT, the deterministic sparse FFT for short-support vectors, the three-dimensional convolutional FFT for molecular electrostatics, and the non-uniform FFT used in radio interferometric visibility simulation each achieve their asymptotic improvements by mechanisms that classical complexity counts simply do not see. An NFFT with a Kaiser-Bessel window appears to run in $O(N\log N + M)$ time, but only after a lookup table of precomputed window values has been built and stored, and the size of that table is what controls the achievable accuracy. A deterministic sparse FFT for vectors of support m achieves $O\big(m\log m\log(N/m)\big)$ arithmetic, where $m \ll N$, by recursively folding the signal and paying for the fold with structured selection matrices that never appear in a butterfly count. A three-dimensional Born-radius computation on a biomolecular grid is dominated, in practice, not by the arithmetic of its $O(N^3\log N)$ FFT but by the memory required to resolve the molecular surface on a sufficiently fine lattice.

The thesis of this review follows from these observations. Modern high-speed FFT architectures conceal large constants inside memory allocation, inside the structure of intermediate matrices, and inside the depth of precomputed tables. Two algorithms with apparently identical arithmetic complexity can differ by an order of magnitude in run-time once memory traffic and precomputation overhead are accounted for. Comparing them on multiplication counts alone produces misleading rankings.

The remainder of the paper develops this argument across four representative algorithmic families. Section 2 re-examines the classical accounting on its own terms, so that the contrast with later sections is exact. Subsequent sections then track how that accounting breaks down as the data geometry departs from a uniform power-of-two grid.

2 The Cost of Precomputation in Non-Uniform Grids

The discrete Fourier transform at nonequispaced nodes generalises the standard DFT to an arbitrary sampling set $X = \{x_j \in \mathbb{T}^d : j = 0, \dots, M-1\}$ together with a multi-degree $N = (N_0, \dots, N_{d-1})^{\mathsf{T}} \in 2\mathbb{N}^d$. Given Fourier coefficients $\hat{f}_k \in \mathbb{C}$ indexed by $k \in I_N$, the task is to evaluate the trigonometric polynomial

$$f(x_j) = \sum_{k \in I_N} \hat{f}_k \, e^{-2\pi i k x_j}, \qquad j = 0, \dots, M-1, \tag{1}$$

where $I_N = \prod_{j=0}^{d-1}[-N_j/2, N_j/2-1] \cap \mathbb{Z}$ is the multidimensional index set, and $N_\pi := N_0 \cdots N_{d-1}$ counts the total number of Fourier coefficients[2]. Written as a matrix-vector product $f = A\hat{f}$ with $A := (e^{-2\pi i k x_j})_{j;k}$, the direct evaluation of 1 takes $O(MN_\pi)$ arithmetic operations and stores no entries of A at all. Its cost lies elsewhere: in MN_π separate evaluations of the complex exponential, which on most hardware are substantially more expensive than the multiplications they replace.

Kunis and Potts establish that the approximate non-equispaced FFT computes the same map in

$$C_{\text{NFFT}} = O(N_\pi \log N_\pi + M) \tag{2}$$

floating-point operations, where the implicit constant of proportionality depends, in theory, solely on the prescribed target accuracy and on the space dimension d[2] . Read against the classical $O(N\log_2 N)$ accounting from Section 1, equation 2 appears to settle the matter in one stroke. The logarithmic factor sits on the equispaced grid; the dependence on the nonequispaced sample count M is strictly linear.

That reading is misleading once the constant attached to the M term is examined.

The construction behind 2 is a three-stage factorisation. For an oversampling factor $\sigma > 1$ and an FFT length $n = \sigma N \in \mathbb{N}^d$, one fixes a window function $\varphi \in L^2(\mathbb{R})$ that is simultaneously well localised in space and in frequency[2]. The NFFT then reads in matrix form

$$A\hat{f} \approx B\,F\,D\,\hat{f}, \tag{3}$$

where D is the $n_\pi \times N_\pi$ deconvolution matrix carrying the inverse Fourier-transformed window, F is the d-variate Fourier matrix of size $n_\pi \times n_\pi$, and B is the real $M \times n_\pi$ sparse matrix

$$B := \left(\tilde{\varphi}\left(x_j - n^{-1} \odot l\right) \cdot \chi_{I_{n,m}(x_j)}(l)\right)_{j=0,\ldots,M-1;\, l \in I_n}, \tag{4}$$

truncated to the index set $I_{n,m}(x_j) := \{l \in I_n : n \odot x_j - m\mathbf{1} \le l \le n \odot x_j + m\mathbf{1}\}$, where $\odot$ denotes element-wise multiplication. The cutoff parameter m controls the support width; each row of B contains exactly $(2m+1)^d$ nonzero entries. For the routine parameter choice $d = 2$ and $m = 4$, every row already carries 81 nonzero values[2].

The truncation in 4 is what makes the bound 2 attainable, and it is also what hides the algorithm's real cost. The combined aliasing and truncation error obeys

$$\left|f(x_j) - s_j\right| \le C_{\sigma,m} \,\|\, \hat{f} \,\|_1, \tag{5}$$

with $C_{\sigma,m}$ decaying exponentially in m. For the dilated Gaussian window of shape parameter $b = 2\sigma m/(2\sigma - 1)\pi$, the explicit estimate is

$$C_{\sigma,m}^{\text{Gauss}} = 4\, e^{-m\pi\left(1 - 1/(2\sigma-1)\right)}, \tag{6}$$

and analogous bounds hold for the B-spline, sinc, and Kaiser-Bessel windows[2]. To reach a target accuracy ε, the cutoff therefore scales as $m \sim \log(1/\varepsilon)$. The convolution step in turn performs $(2m+$

$1)^d M$ flops, linear in M as advertised, but each of those flops requires a window value $\tilde{\varphi}\left(x_j - n^{-1} \odot l\right)$ that has to come from somewhere.

The relevant engineering question is not how many multiplies the algorithm performs but where the window data resides. Kunis and Potts tabulate the available strategies, each trading storage against per-flop overhead. The principal options are summarised in Table 1.

Scheme	Memory (real numbers)	Extra arithmetic per node
no precomputation	0	$(2m+1)^d$ window evaluations
FG_PSI	0	$O(dm)$
PRE_LIN_PSI	dK	$O(dm)$
PRE_FG_PSI	dM	$O(dm)$
PRE_PSI	$d\,m\,M$	—
PRE_FULL_PSI	$m^d M$	—

Table 1: Memory and arithmetic overhead of the convolution step in the NFFT, after Kunis and Potts . Of these schemes, only the lookup-table method PRE_LIN_PSI has a footprint independent of the sampling set $\{x_j\}$[3].

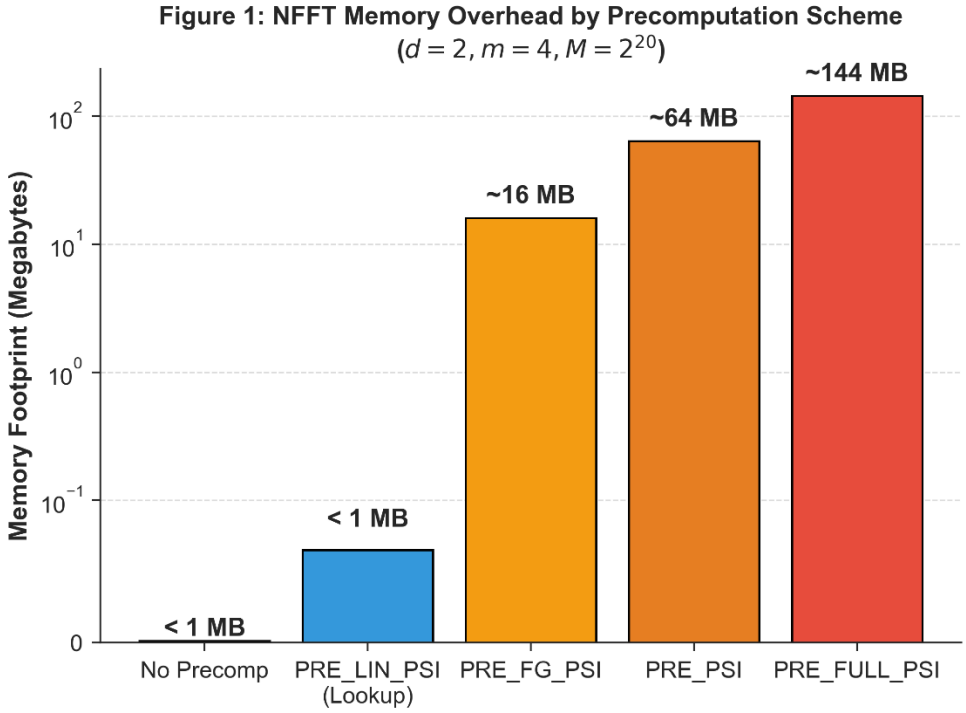

Figure 1: NFFT memory overhead for various precomputation schemes ($d = 2, m = 4, M = 2^{20}$). The logarithmic y-axis reveals multi-order-of-magnitude memory differences hidden within the linear $O(M)$ asymptotic bound.

The asymptotic notation in 2 hides differences of several orders of magnitude. For a univariate problem with $m = 4$ and $M = 2^{20}$ nodes, PRE_FULL_PSI — which gives the fastest convolution step by a factor of 5 to 100 over the no-precomputation default — already consumes roughly 144 MByte purely for the entries of B and their indices[2]. The "linear-in-M" claim carries a constant of $(2m+1)^d \cdot$ sizeof(double) bytes per node when full precomputation is selected. In three dimensions with a moderate cutoff $m = 6$, the per-node constant alone is on the order of $13^3 \approx 2200$ doubles.

A detail noted in passing by Kunis and Potts deserves more weight than the surrounding discussion of asymptotic rates. Of the precomputation strategies in Table 1, only the lookup-table method PRE_LIN_PSI has a memory cost dK that does not scale with the sampling set $\{x_j\}$[2] . Every other scheme allocates per-node storage, so doubling the input automatically doubles the table. The lookup approach escapes this coupling because the window is sampled once on a fine equispaced grid of K points and then linearly interpolated; the interpolation error decays as $1/K^2$, so $K \sim \varepsilon^{-1/2}$ values suffice for accuracy ε. Empirically, $K = 2^{12}$ entries are reported to deliver single precision 10^{-8} regardless of the problem size[2].

The relevance of this distinction to the thesis of Section 1 is direct. The error bound 5 is an L_1 statement about coefficient amplitudes; standard NFFT analyses additionally report an L_2 relative accuracy $E_2 = \| f - s \|_2 / \| f \|_2$[2] . Neither bound carries any information about whether the underlying window evaluation cost the implementation 0 bytes per node, dm bytes per node, or m^d bytes per node. Two NFFTs may attain identical E_2 at identical flop count and yet differ by two orders of magnitude in their physical memory footprint. The asymptotic complexity 2 cannot separate the two, and the stability theory does not attempt to.

The point is not that the NFFT is poorly engineered; the precomputation hierarchy of Kunis and Potts is the correct response to a memory-bound problem. The point is that the cost has moved. An arithmetic problem with a small memory footprint has become a memory-allocation problem with a small arithmetic footprint, and the standard complexity statement 2 captures only the second half of that trade.

3. Multi-Dimensional Bottlenecks in High-Density Solvers

The memory-versus-arithmetic trade examined in Section 2 sharpens as dimensionality rises. A one-dimensional NFFT pays a constant of $(2m+1)$ per node; a three-dimensional one pays $(2m+1)^3$. The same scaling appears in two problem classes that, on the surface, look unrelated: dense biomolecular grids for implicit solvation, and dense interferometric arrays for low-frequency radio astronomy. Both replace an apparently quadratic direct evaluation with an FFT plus interpolation, and both pay for the speed-up in physical memory that the asymptotic complexity does not name.

Cai, Xu, and Baumketner attack the generalised Born (GB) problem, in which the effective Born radius R_i of each atom i in a solute molecule must be obtained from a volume integral of a singular kernel over the molecular exterior[3]. Treated atom by atom, this is a grid calculation of complexity $O(MN^3)$ for M atoms on a domain discretised at N^3 lattice points; even the surface-integral reformulation cannot push the cost below $O(MN^2)$, which remains prohibitive for proteins of practical size[3].

$$\Phi(\mathbf{r}) = \int_{\Omega_{\text{in}}} G(\mathbf{r} - \mathbf{r}')\, d\mathbf{r}', \tag{7}$$

where Ω_{in} denotes the interior volume of the solute molecule. This convolution can be evaluated on the entire lattice through a pair of three-dimensional FFTs at a cost of $O(N^3 \log N)$[3]. Off-grid atomic positions are recovered by interpolation from nearby lattice values, contributing a further $O(M)$. The combined scaling reads

$$C_{\text{GB-FFT}} = O(N^3 \log N + M), \tag{8}$$

and, as the authors stress, the grid parameter N depends only on the geometry of the biomolecule and on the spectral decay of the smoothing kernel; it does not scale with M .

The independence of N from M is the formal reason equation 8 reads as linear in the atom count. The practical reason it cannot be treated as free, however, sits in the constant on the N^3 term. For protein 3GB1, the reference calculation used a 128^3 grid on a domain $[-32\,\text{Å}, 32\,\text{Å}]^3$, corresponding to a lattice spacing $h = 0.5\,\text{Å}$; the larger protein 1OCA was treated similarly[3]. At that resolution, the molecular surface is only marginally resolved. Accuracy studies on a spherical solute showed that the smoother width a and the lattice spacing h interact strongly: reducing h from 0.125 to 0.03125, a 64-fold increase in grid volume, brought the relative error in R_i from order 5% down to below 1%, but only when a was chosen in coordination[3].

This is where the cache problem enters the analysis. A 128^3 array of double-precision complex numbers occupies $128^3 \times 16 \approx 32$ MByte before any auxiliary storage; a 256^3 array occupies 256 MByte; a 512^3 array occupies 2 GByte. The reference hardware in was a notebook with 2 GByte of total memory, which placed a hard ceiling on the achievable N. The grid spacing h is therefore not a free knob to be tightened until the discretisation error is satisfactory. It is bounded below by what fits in RAM. The smoother width a exists precisely to relax this bound, allowing a coarser grid to deliver acceptable accuracy by softening the kernel until its spectral decay is fast enough that fewer Fourier modes are needed[3].

Three-dimensional caching effects compound the problem. A 128^3 complex grid exceeds typical L3 cache capacity by orders of magnitude, so each FFT pass tends to be bandwidth-bound rather than arithmetic-bound; wall-clock time tracks DRAM throughput, not multiplier count. The reported $5\times$ to $14\times$ speed-ups of the FFT method over the grid-based reference (Tab. 4 of[3]) hold within the parameter window where the grid fits comfortably in memory. Stepping outside that window, towards higher resolution or larger proteins, pushes the algorithm onto a grid the host hardware cannot hold.

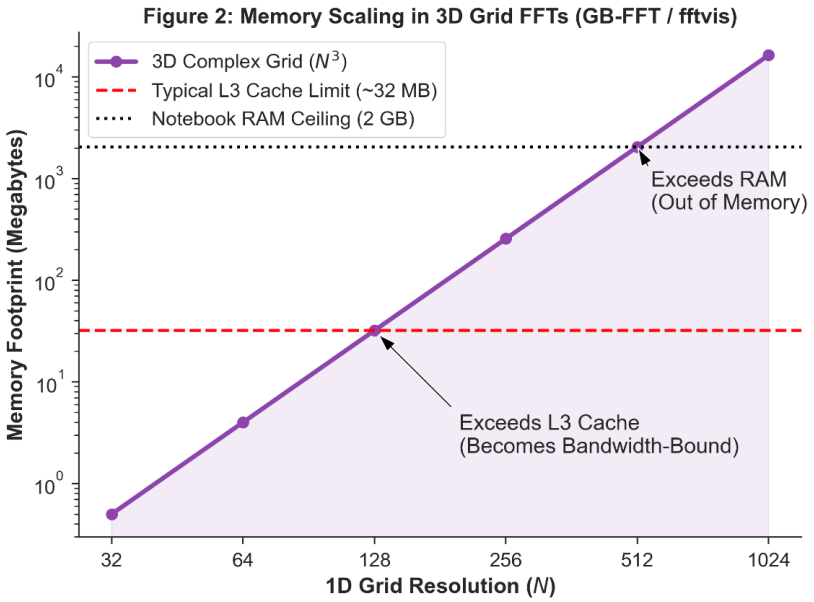

Figure 2: Memory scaling of 3D grid FFTs (fftvis / GB-FFT) versus 1D resolution N. Crossing hardware thresholds (L3 cache, RAM) forces a shift from compute-bound to bandwidth- and memory-bound regimes.

The visibility-simulation problem in 21-cm cosmology presents the same trade in a different physical setting. The radio interferometer measurement equation under the point-source approximation requires evaluating, for each baseline (i,j) and each time-frequency snapshot, a sum of complex exponentials over N_{sources} sky components and N_{bls} baselines. Direct evaluation costs $O(N_{\text{sources}}N_{\text{bls}})$[4] . The reference simulator matvis reduces this cost via a per-antenna outer product of size $N_{\text{sources}} \times N_{\text{ants}}$, exchanging arithmetic redundancy for the storage of an explicit intermediate matrix[4].

The fftvis algorithm of Cox and colleagues replaces this matrix path with a Type-3 non-uniform FFT through the finufft library. Visibilities at every baseline are computed simultaneously from a single NUFFT call per time-frequency channel, and the cost scales as

$$C_{\text{fftvis}} = O\left(N_{\text{sources}}w^d + N_{\text{grid}}\log N_{\text{grid}} + N_{\text{bls}}w^d\right), \tag{9}$$

where w is the gridding-kernel half-width, d is the dimensionality of the antenna coordinate space (two for planar arrays, three otherwise), and $N_{\text{grid}} = \prod_{i=1}^{d} n_i$ is the size of the oversampled intermediate grid[4]. The dimension in direction i is set by

$$n_i = \left\lceil \frac{4\sigma\, b_{i,\max}\, \nu}{c} + w \right\rceil, \tag{10}$$

with $b_{i,\max}$ the maximum baseline extent in direction i, ν the observing frequency, c the speed of light, and σ an oversampling factor[4].

Reading equation 10 carefully is what exposes the underlying trade. The intermediate grid scales linearly with the longest baseline in wavelength units, $u_{\max} = b_{\max}\nu/c$, independently of how many antennas the array contains. For a compact, densely packed array the grid stays small and the NUFFT path dominates; for a sparse array with long baselines the grid grows enormous, and the NUFFT path becomes both

compute-bound, through the $N_{\text{grid}}\log N_{\text{grid}}$ FFT, and memory-bound, through the $4\sigma N_{\text{grid}}$ allocation that finufft requires internally[4]. The projected benchmark table in[4] makes the trade explicit. For the SKA-LOW layout, in which $u_{\max}$ reaches the thousands of wavelengths, fftvis requires on the order of 5.8×10^5 seconds for a diffuse sky model at 200 MHz; matvis on the same problem requires 1.3×10^8 seconds, but its memory footprint behaves more gracefully because the outer-product matrix scales with array population, not with array extent.

The memory accounting in fftvis is dominated by exactly this intermediate grid. Cox and colleagues enumerate six contributing arrays: source flux, output visibilities, raw beam map, interpolated beam values, source coordinates, and the NUFFT intermediate grid. The last is the one that explodes with $u_{\max}$[4]. In matvis, by contrast, memory is dominated by the $O(N_{\text{sources}}N_{\text{ants}})$ outer-product matrix, which can be chunked along the source axis to almost arbitrary reduction[4].

The two simulators therefore occupy complementary corners of a memory-bound trade. Dense arrays, in which $N_{\text{ants}}/\sqrt{u_{\max}}$ is large, are where fftvis is both faster and lighter, by up to two orders of magnitude. Sparse arrays with long baselines are where the intermediate FFT grid swallows the algorithm's advantage, and matvis, or a Type-1 NUFFT on a regularised baseline grid, becomes preferable[4].

The structural parallel with Cai et al. is exact. In both cases the arithmetic complexity carries a logarithmic factor on a grid whose size is fixed by a physical-extent parameter, the molecular diameter for the Born-radius problem and the longest baseline for the visibility problem. In both cases the grid must be allocated in full before the FFT can be performed, and the memory required for that allocation determines whether the algorithm outruns its direct competitor or falls behind. The N^3 in equation 8 and the N_{grid} in equation 9 are the same hidden constant in different clothes.

4. Asymptotic Trade-offs and the Complexity Shift

The algorithms surveyed in Sections 2 and 3 share a common feature. They assume little about the structure of the data being transformed and pay for that generality in memory. Plonka and Wannenwetsch[5] take the opposite line. They impose a structural condition on the input, namely a real non-negative signal vanishing outside an interval of length $m \ll N$, and recover dimensional savings in arithmetic that the NFFT and three-dimensional FFT classes cannot approach.

Their algorithm recovers a vector $\mathbf{x} \in \mathbb{R}_+^N$ from its discrete Fourier transform $\hat{\mathbf{x}} = F_N\mathbf{x} \in \mathbb{C}^N$ through a divide-and-conquer scheme built on successive periodisations. At iteration j the periodised vector $\mathbf{x}^{(j)}$ has support length m_j, and the construction exploits the observation that m_j is bounded by the realised full-vector support m at every level. The resulting arithmetic complexity is

$$\mathcal{C}_{\text{sparse}} = O\big(m\log m\log(N/m)\big), \tag{11}$$

with at most $O\big(m\log(N/m)\big)$ Fourier samples consumed[5]. For $\log m$ bounded, the bound in 11 sits at $O(m\log N)$, well below the $O(N\log N)$ lower bound that Morgenstern established for general length-N FFTs[5]. The algorithm requires no a priori knowledge of m. It adapts at each periodisation level, automatically recognises a short support during the computation itself, and reverts to a standard radix-2 FFT of cost $O(N\log N)$ when the input is effectively full-support[5].

Three features of 11 stand out against the bounds collected in the previous two sections.

The first is the absence of any per-node memory constant. No factor of $(2m+1)^d$ multiplies the dominant term. No oversampled intermediate grid is allocated. No precomputed window table of size dK or dmM has to be filled before the recursion can begin. Memory overhead is bounded by the size of the input together with the support indices at each periodisation level, which Plonka and Wannenwetsch bound by $O(m_{j-1})$ at iteration j[5]. The asymptotic statement in 11 is, on this point, honest in a way that the NFFT and GB-FFT bounds in equations 2 and 8 are not.

The second is the price of the prior. Real non-negativity is enforced during reconstruction through a threshold parameter T. Under additive noise of bounded amplitude $|\varepsilon_k| \le \delta$, T must be tuned to the noise floor in order to separate recovered components from spurious oscillation. Numerical experiments at signal-to-noise ratios between 10 and 50 dB, on vectors of length $N = 2^{15}$ with support $m = 15$, confirm that the algorithm remains numerically stable, but the threshold has to be reselected for each noise level[5]. The bound in 11 is therefore paid for in a different currency from the NFFT's window precomputation. It is paid for in the strength of the structural assumption placed on the input.

The third is the contrast with earlier sparse-FFT constructions. Randomised schemes such as those of Hassanieh et al. and Pawar-Ramchandran achieve $O(k\log N)$ or even $O(k\log k)$ for k-sparse signals, but only with high probability and with no sublinear method available to check the result[5]. Deterministic combinatorial schemes based on prime-length FFTs and the Chinese remainder theorem are correct but carry polynomial costs in $\log N$ and k that pay off only for very large N and strong sparsity. Prony-based methods incur $O(k^3)$ through the singular value decomposition of structured matrices and are competitive only for very small k[5]. The Plonka-Wannenwetsch construction is deterministic and self-adaptive; it requires no k or m prior, and this absence is what classical complexity counts could not capture. The cost depends on the realised structure of the input, not on N alone.

The picture across the algorithmic families covered in this review is summarised in Table 2.

Figure 3: The Asymptotic Trade-off Landscape

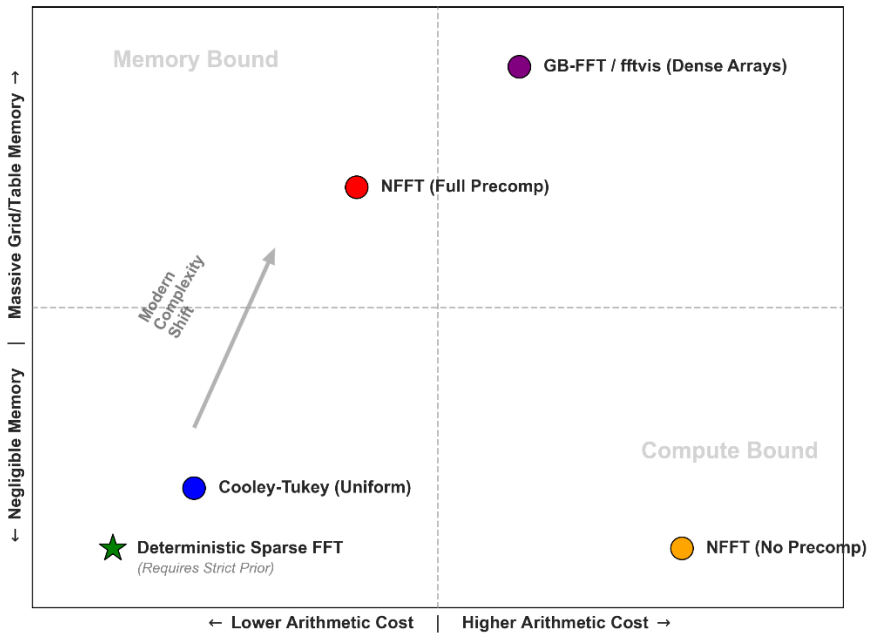

Figure 3: The asymptotic trade-off matrix. Five FFT classes ordered from the classical uniform-grid baseline, through non-uniform and high-dimensional schemes, to the structurally constrained deterministic sparse case. The third column reports the bound as stated by the cited paper; the fourth identifies the cost that the asymptotic notation does not name.

Algorithm class	**Target data structure**	**Asymptotic time complexity**	**Primary memory or stability penalty**
Cooley-Tukey / split-radix[1]	Uniform 1-D grid of length $N = 2^n$	$O(N\log_2 N)$	Twiddle-factor tables of $\sim N$ words; bit-reversal permutation; addition count grows with radix.
NFFT[2]	Non-equispaced nodes in $\mathbb{T}^d$, multi-degree N, M samples	$O(N_\pi \log N_\pi + M)$	Window precomputation: dK (lookup table), dmM (tensor product), or $m^d M$ (full matrix B); ≈ 144 MByte at $m = 4$, $M = 2^{20}$.
GB-FFT[3]	3-D Cartesian lattice of N^3 points containing M atoms	$O(N^3 \log N + M)$	Full N^3 complex grid: 32 MByte at $N = 128$, 2 GByte at $N = 512$; exceeds L3 cache for $N \gtrsim 64$.
NUFFT visibility (fftvis)[4]	Non-uniform u,v-plane baselines, N_{src} sky sources, N_{bls} baselines	$O\big(N_{\text{src}} w^d + N_{\text{grid}} \log N_{\text{grid}} + N_{\text{bls}} w^d\big)$	Intermediate FFT grid of size $4\sigma N_{\text{grid}}$; scales linearly with $u_{\max}$; dominates memory for sparse arrays with long baselines.
Deterministic sparse FFT[5]	Real, non-negative $\mathbf{x} \in \mathbb{R}_+^N$ with short support $m \ll N$	$O\big(m \log m \log(N/m)\big)$	Negligible additional storage; stability tied to threshold parameter T and the input signal-to-noise ratio.

Table 2: The Asymptotic Trade-off Landscape. As data geometry departs from uniform grids, computational costs migrate from arithmetic into memory and precomputation (Y-axis) or demand strict structural priors.

Reading Table 2 along the time-complexity column gives the conventional narrative of FFT progress: $O(N\log N)$ for the uniform case, the same up to log factors for non-uniform cases in higher dimensions, and a sublinear bound for structured sparse inputs. Reading the same table along the penalty column tells a different story. The Cooley-Tukey case carries a small and well-understood memory constant. The NFFT, the GB-FFT, and the NUFFT-visibility cases each carry penalty terms that grow with dimension or with physical extent and that are, in absolute terms, far larger than the arithmetic count alone would suggest. The deterministic sparse case has the smallest memory penalty of any entry in the table, but its applicability is restricted to inputs that satisfy a structural condition the other classes do not require.

The complexity shift announced in Section 1 is therefore not a single migration from arithmetic to memory. It is a redistribution. As the data geometry departs from the uniform grid, the cost migrates into the memory column. As the structural assumption on the data is strengthened, the cost migrates back out of memory and into the question of applicability, that is, into whether the assumption holds for any given problem instance.

5. Conclusion

The arithmetic-cost metric that organised FFT research for three decades no longer ranks competing algorithms in the order their wall-clock times deliver. The five algorithmic families surveyed here distribute their cost across different combinations of arithmetic and memory load, and the dominant term varies across the table. Comparing them on multiplication counts alone is not merely a partial picture. On contemporary hardware, for several of the entries in Table 2, it is an inverted one.

Section 2 showed that the NFFT bound $O(N_\pi \log N_\pi + M)$ carries inside the M term a window-precomputation cost ranging from negligible to several hundred megabytes for the same accuracy, depending on which of the precomputation flags collected in Table 1 is selected. Section 3 extended the observation to three dimensions: the GB-FFT for biomolecular solvation and the fftvis simulator both achieve $O(N^3 \log N)$, or its NUFFT equivalent, on a grid whose size is set by a physical-extent parameter and that must be allocated in full before any FFT can run. Section 4 placed against these the deterministic sparse construction of Plonka and Wannenwetsch, where the memory column collapses but only at the cost of imposing a non-negativity and short-support condition on the input.

The thesis announced in Section 1 is upheld in each of these cases. Hidden constants live in memory allocation, in lookup tables, in oversampled grids, and in the sparse selection matrices that classical arithmetic counts cannot see. Two algorithms with identical $O(N\log N)$ bounds can differ by orders of magnitude in their physical footprint and, on memory-bound hardware, in their wall-clock time.

The consequence for algorithmic evaluation is more pointed than a general call for better metrics. The two reporting conventions that dominate the field, namely asymptotic flop count and the L_2 relative error $E_2 = \| f - s \|_2 / \| f \|_2$, appear jointly insufficient for the algorithms now in active use. Neither carries information about memory traffic, cache behaviour, oversampling factors, or the size of the precomputation tables that the reported accuracy actually required. A complete evaluation of an FFT

algorithm on contemporary hardware seems to demand at minimum three quantities: an arithmetic count with its constant named, a memory footprint that includes precomputation, and an indication of how that footprint scales with the structural parameters of the problem.

A methodological corollary follows. The community-standard practice of treating O-notation as a sufficient summary of cost has been carried over from the era when arithmetic dominated, and it now produces rankings that the implementations themselves contradict. The benchmark tables of Kunis and Potts and of Cox and colleagues both report identical asymptotic complexities for schemes that differ in run-time by factors between 5 and 100. The discrepancy is not noise in the implementation. It is information that the asymptotic statement was not constructed to carry. Future work on FFT-class algorithms would, on our reading, benefit from reporting memory penalties at the same level of precision as flop counts, and from treating the choice of precomputation scheme as part of the algorithm rather than as a detail of its implementation.

There is no obvious resolution to the trade itself. The Cooley-Tukey accounting was complete because the hardware of its era priced arithmetic operations and memory access at comparable rates[1]. The hardware on which modern FFTs run prices the two very differently, with memory bandwidth typically two orders of magnitude scarcer than multiplier throughput per unit of silicon. The algorithms have responded by trading away the cheaper resource for access to the more expensive one. The metric for evaluating them has not yet caught up, and it is the lag between the two that this review has attempted to make explicit.

REFERENCES:

[1] P. Duhamel and M. Vetterli, “Fast Fourier transforms: A tutorial review and a state of the art,” *Signal Processing*, vol. 19, no. 4, pp. 259–299, 1990.

[2] S. Kunis and D. Potts, “Time and Memory Requirements of the Nonequispaced FFT,” *Sampling Theory in Signal and Image Processing*, vol. 7, no. 1, pp. 77–100, 2008.

[3] W. Cai, Z. Xu, and A. Baumketner, “A new FFT-based algorithm to compute Born radii in the generalized Born theory of biomolecule solvation,” *Journal of Computational Physics*, vol. 227, no. 24, pp. 10162–10177, 2008.

[4] T. A. Cox et al., “fftvis: a non-uniform Fast Fourier Transform based interferometric visibility simulator,” *RAS Techniques and Instruments*, vol. 4, no. 1, pp. 1–20, 2025.

[5] G. Plonka and K. Wannenwetsch, “A sparse fast Fourier algorithm for real nonnegative vectors,” *Journal of Computational and Applied Mathematics*, vol. 321, pp. 532–539, 2017.