Scope

This note sketches how two-for-one real ↔ complex FFT algos 2 are used for circle group LDEs 1 in our code.

Notation

j: $\sqrt{-1}$
N: Size of witness columns
H: Order-N subgroup of the circle group
ω: Generator of H
Following 2's convention, lowercase arrays are evals indexed by n and uppercase arrays are monomial coeffs indexed by k, e.g.

  $a[n], b[n]$: Real evals for two witness columns on domain H
  $A[k], B[k]$: Monomial coeffs that interpolate $a[n]$, $b[n]$

Polynomials are lowercase letters with the arg in parentheses, e.g. $a(x)$, $b(x)$.
FFT and IFFT definitions are the opposite of 2's convention:

  FFT: Monomial coeffs → evals on H
  IFFT: Evals on H → monomial coeffs

FFTs and IFFTs here are always size N.

Two-for-one

The two-for-one trick performs two IFFTs for real-valued input arrays a, b with a single complex-valued IFFT, by packing them as coeffs of a single complex array and IFFTing the complex array.

$$\begin{aligned} z[n] &\equiv a[n] + jb[n] \\ Z &\equiv \mathrm{IFFT}(z) \\ Z[k] &= A[k] + jB[k] \quad \text{by linearity} \end{aligned}$$

By conjugate symmetry of IFFT for real-valued inputs, $A[k] = A^*[N-k]$ and $B[k] = B^*[N-k]$. Therefore (if desired) we can recover $A[k]$ and $B[k]$ via:

$$\begin{aligned} A[k] &= \frac{Z[k] + Z^*[N-k]}{2} \\ B[k] &= -j\frac{Z[k] - Z^*[N-k]}{2} \end{aligned}$$

Real-valued circle group LDEs

Let $a(x) = \sum_{k=0}^{N-1} A[k]x^k$, the polynomial that interpolates a on H. Let τ be a coset offset factor taken from the circle group.

1 shows the evals of $a_{\text{shifted}}(x) \equiv \tau^{-\frac{N}{2}}(a(x) - A[0])$ on the coset domain $\tau \cdot H$ are real. These are the evals we'd like to compute and commit to. We compute them via the usual LDE procedure: First, define

$$a_{lde}(x) \equiv \sum_{k=0}^{N-1} A_{lde}[k]x^k \quad \text{where} \quad A_{lde}[0] = 0, \;\; A_{lde}[k > 0] = \tau^{k-\frac{N}{2}} A[k]$$

Coeffs of $a_{lde}(x)$ are the coeffs of $a_{\text{shifted}}(x)$ multiplied by τ^k, such that evals of $a_{lde}(x)$ on H match evals of $a_{\text{shifted}}(x)$ on $\tau \cdot H$. $a_{lde} \equiv \mathrm{FFT}(A_{lde})$ then gives these desired evals.

Efficient two-for-one LDEs

Step 1: Pack z via $z[n] = a[n] + jb[n]$ and compute $Z = \mathrm{IFFT}(z)$. Don't bother to recover A and B. We don't explicitly need them.[1]

[1] 1 says we do need $A[0]$ and $B[0]$, so the prover can later compute (complex) LDE-domain evals of $a(x)$ and $b(x)$ from (real) committed values a_{lde} and b_{lde}. For example, $a(\tau w^n) = A[0] + \tau^{\frac{N}{2}} a_{lde}[n]$. But luckily, $A[0]$ and $B[0]$ are the (real) averaged sums of a and b, so we can recover them as $\mathrm{Re}(Z[0])$, $\mathrm{Im}(Z[0])$.

Step 2: Read each $Z[k]$ and compute[2] $Z_{lde}[0] = 0$, $Z_{lde}[k > 0] \equiv \tau^{k-\frac{N}{2}} Z[k]$. Note that

$$\begin{aligned}
Z_{lde}[0] &= A_{lde}[0] + jB_{lde}[0] = 0 \\
Z_{lde}[k > 0] &= \tau^{k-\frac{N}{2}} Z[k] \\
&= \tau^{k-\frac{N}{2}} (A[k] + jB[k]) \\
&= \tau^{k-\frac{N}{2}} A[k] + j\tau^{k-\frac{N}{2}} B[k] \\
&= A_{lde}[k] + jB_{lde}[k]
\end{aligned}$$

A_{lde} and B_{lde} are themselves complex-valued, but that's fine. The FFT will disentangle them.

Step 3: Compute $z_{lde} = \mathrm{FFT}(Z_{lde})$. Note that

$$\begin{aligned}
z_{lde} &= \mathrm{FFT}(A_{lde} + jB_{lde}) \\
&= a_{lde} + jb_{lde}
\end{aligned}$$

Step 4: a_{lde} and b_{lde} are real, so

$$\begin{aligned}
a_{lde}[n] &= \mathrm{Re}(z_{lde}[n]) \\
b_{lde}[n] &= \mathrm{Im}(z_{lde}[n])
\end{aligned}$$

References

[1] Ulrick Haböck, Daniel Lubarov, Jacqueline Nabaglo. *Reed-Solomon codes over the circle group.* https://eprint.iacr.org/2023/824.pdf

[2] Robin Scheibler. *Real FFT Algorithms* https://www.robinscheibler.org/2013/02/13/real-fft.html

[2]With precomputed power-tables, this is easy in a kernel, and no more expensive than applying prefactors for an ordinary LDE.