Computation of Zero Points with Weierstraß Iteration in OpenCL1.2

University of Regensburg

Thomas Karl

July 1, 2021

1 Setup

The program computes complex zero points of arbitrary complex polynomials numerically using Weierstraß Iteration. The procedure is accelerated on GPUs with OpenCL 1.2. Inputs are the number of polynomials n and its degree d. Let p be the number of cores. The number of polynomials n_p that can be computed in parallel is $n_p = \lfloor p/d \rfloor$. If $dn_p \neq p$, some cores remain idle.

Each core computes an initial guess and randomly a complex coefficient for the polynomial. Real- and imaginary part lie uniformly distributed between two parameters that where also given as user input. The zero points have to be complex and lie in the initially on a circle in the complex plane. The radius is the half distance between the parameters.

Each core improves its k-th zero point x_k using the iteration,

$$x_k^{i+1} = x_k + \frac{\text{polynomial}(x_k^i)}{\prod_{j=1;j\neq k}^{d}(x_k^i - \zeta_j)} \longrightarrow x_k \qquad i \to \infty \tag{1}$$

ζ_j denote all zero points so far. Since a dynamically change of the ζ_j in each iteration accelerates the convergence, dataraces can be ignored, The iteration breaks after $10d$ runs or if for a a user given $\varepsilon > 0$ the equation

$$\max\left\{\frac{|\text{polynomial}(x)|}{|x|}\right\} \leq \varepsilon \tag{2}$$

holds. This iteration is commonly known as *Weierstraß-Durand-Kerner* Iteration. The zero points are converted in coordinates in a 1080×1080 pixel image and generated on the CPU in *ppm* format. Figure 3 shows an example.

2 Evaluation

For the evaluation of the performance the computation times where recorded and fitted linearly. Figure 1 shows the comparison of computation times on a CPU and a GPU. The GPU was a *Nvidia GTX 1060*, the CPU an *Intel i7 8700K 4.8GHz*. Using 1280 cores we get a speedup with respect to the serial program of roughly 70. The latency due to the data traffic reads about 0.5 seconds.

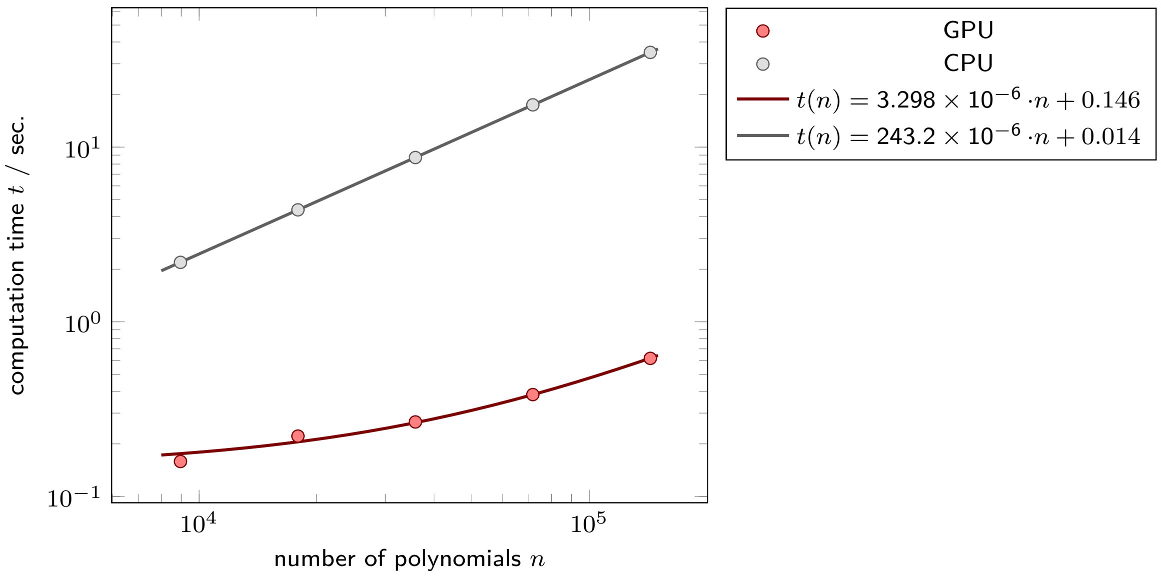


Figure 1: Computation times for certain numbers of polynomials.

In the last step the dependency of the polynomial degree is evaluated (Fig. 2). When the degree is doubled, for the same number of polynomials twice the number of zero points needs to be computed. The number of floating point operations and the iterations increase also with the degree. When the computation times are compared for the same number of zero points $(n \cdot d)$ For degree 8 to 32 the computation time is almost the same for a larger number of polynomials.

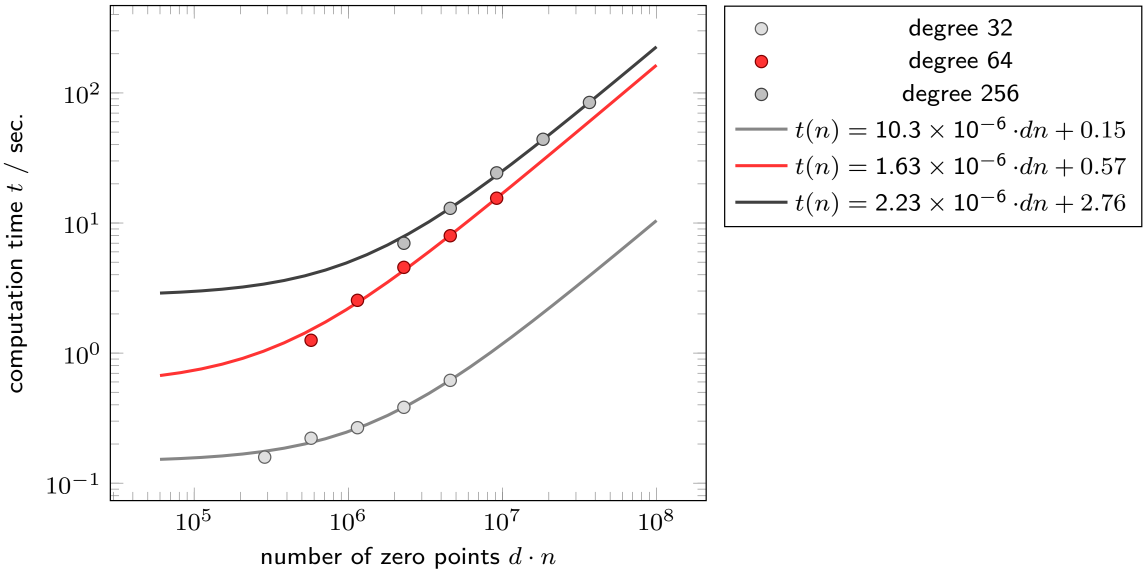
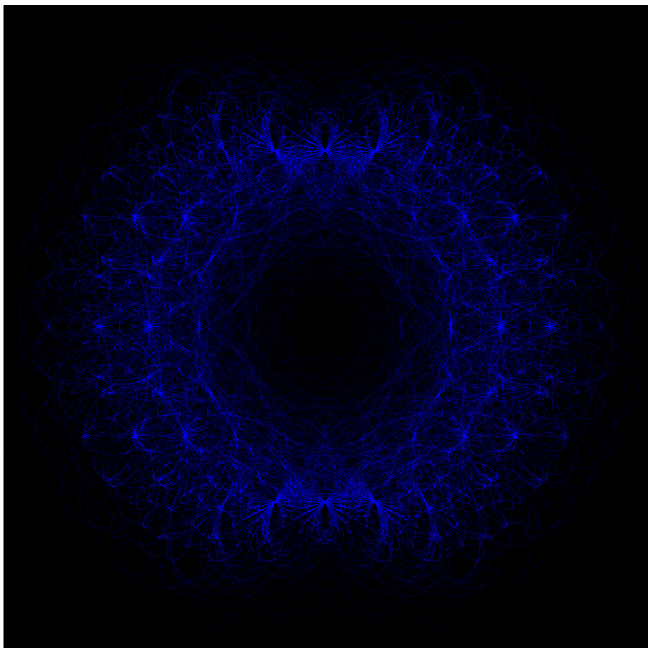
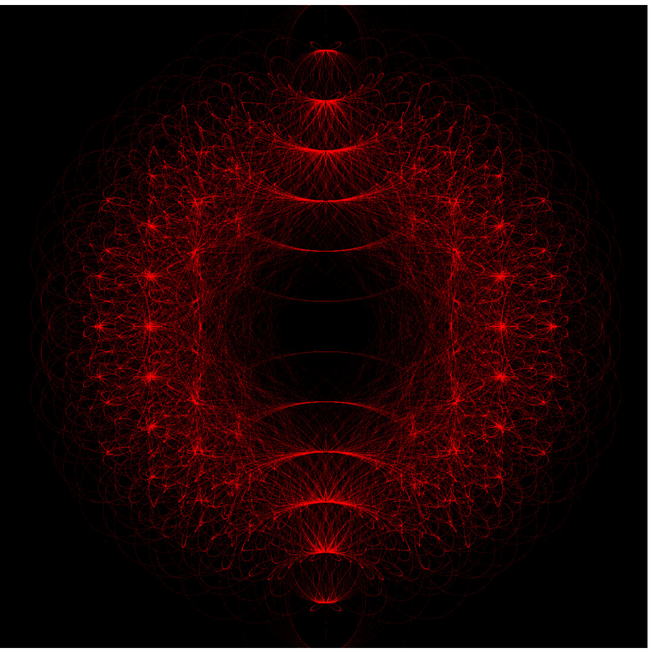


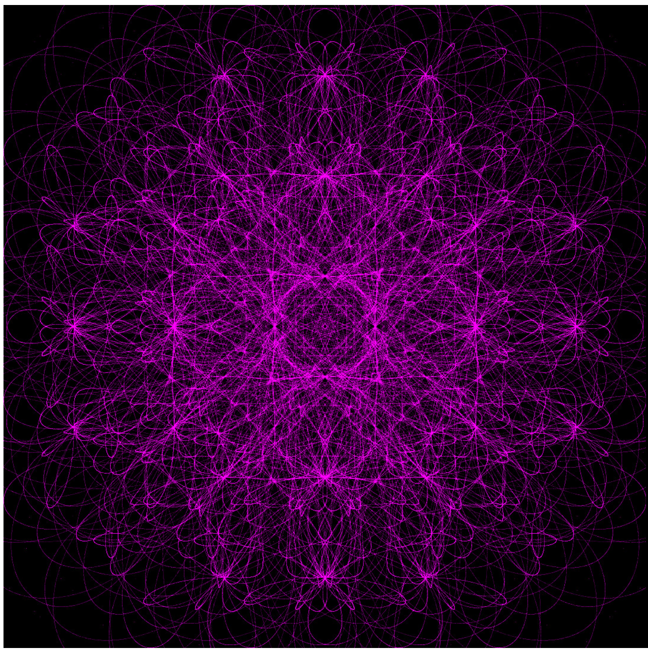
Figure 2: Computation times for certain number of polynomials.



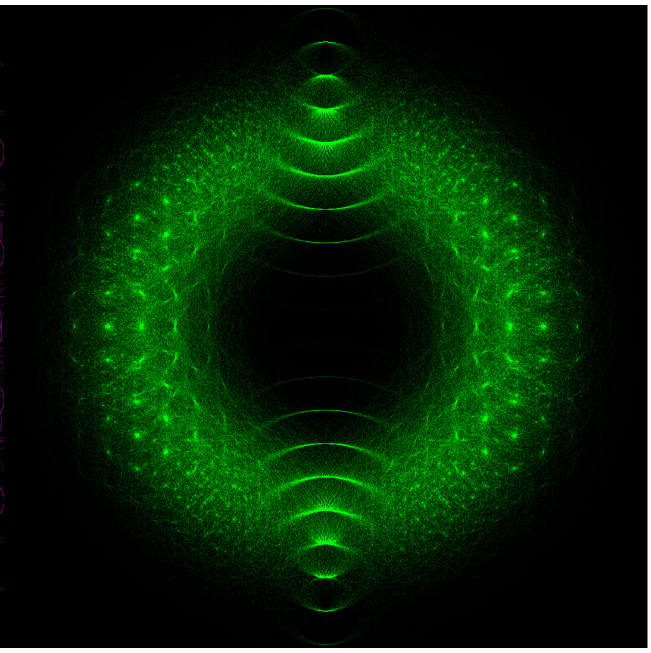
$n = 30 \times 10^6$, $d = 8$, imaginary part 1 or -1



$n = 30 \times 10^6$, $d = 8$, real part 1 or -1



$n = 60 \times 10^6$, $d = 4$, real and imaginary part 1 or -1,



$n = 40 \times 10^6$, $d = 12$, real part 1 or -1

Figure 3: Graphical representation of zero points in complex plane (1080×1080 pixel, $\varepsilon = 0.0001$). The zero points are distributed on a circle with radius one.