

# Analyzing Optimal Hyperparameters for the Radio Interferometric Calibration Method StEFCal

Pieter T. Colesky, Trienko L. Grobler, and Waldo Kleynhans

*Abstract* – The ever-increasing data volume that radio interferometers need to process requires computationally efficient calibration algorithms. StEFCal is a well-known calibration algorithm in the industry. Hyperparameter optimization is a promising approach for enhancing the performance of algorithms in general. In this paper, particle swarm optimization is used to identify better hyperparameters for StEFCal. The hyperparameters mentioned enabled StEFCal to converge in fewer iterations. In addition, the hyperparameter search space is visualized.

## 1. Introduction

A radio interferometer is an array of antennae that work together as a single instrument. In radio astronomy, an interferometer is used to observe and characterize celestial objects visible in the radio spectrum. Various environmental and instrumental factors introduce errors in the signal measured by a radio interferometer. Calibration is the overarching term that is used to describe the process that is followed to try and eliminate errors that are introduced into interferometric observations [1]. Calibration is a highly active research field, and the following papers are particularly noteworthy. Traditionally, calibration is cast into a real optimization problem [2]. However, Smirnov and Tasse [3] reformulated calibration as a complex optimization problem by using Wirtinger derivatives and, in doing so, created a simplistic mathematical framework with which calibration can be achieved. As radio antennas become more advanced, larger amounts of data are being captured. Calibration pipelines contain hyperparameters, and it is becoming more prudent to optimize these hyperparameters for optimal system performance and throughput due to the large data volumes [4]. Yatawatta and Avruch [5] used reinforcement learning to optimize the hyperparameters of a calibration pipeline. Salvini et al. [6] developed the calibration algorithm StEFCal, a well-adopted calibration algorithm that scales quadratically with the number of antennas in the array.

StEFCal has one hyperparameter, the so-called damping factor  $\lambda$ , which is set to 0 during odd iterations

and to 1 in even iterations; consequently, averaging is performed in even iterations. This setup is not guaranteed to be optimal, but it is well-known to produce stable, fast, and accurate results [7]. This paper investigates whether the default hyperparameters commonly used in the literature for the StEFCal algorithm are, in fact, optimal. The paper first describes calibration. Then, it presents the StEFCal algorithm and its hyperparameters. It then discusses various hyperparameter optimization approaches and the simulator used to obtain the main results presented in the paper. The paper concludes with a detailed discussion of the methodology used, the results obtained, and the key findings of this paper.

## 2. Calibration

Each pair of antennas  $pq$  in an array forms a baseline. Therefore, an interferometer with  $N$  antennas has  $B = N(N - 1)/2$  baselines. Each baseline records the (observed) visibility  $d_{pq}$  [8], i.e.,

$$d_{pq} = g_p m_{pq} \bar{g}_q + n_{pq} \quad (1)$$

where  $m_{pq}$  represents the model visibilities,  $g_p$  and  $g_q$  are the complex antenna gain parameters associated with antenna  $p$  and  $q$ , respectively, and  $n_{pq}$  is the thermal noise component, which, in this paper, is assumed to be normally distributed. The bar overhead notation denotes conjugation. Least-squares minimization is a technique often used to solve an overdetermined system of equations, requiring the minimization of the sum of the squared residuals. This is a standard approach in regression analysis. In radio interferometry, least-squares minimization is used to estimate the complex gain vector  $\mathbf{g} = [g_1, g_2, \dots, g_N]^T$  to correct the observed visibilities (i.e., for calibration). In practice, an augmented parameter vector  $\check{\mathbf{g}} = [\mathbf{g}^T, \bar{\mathbf{g}}^T]^T$  is used, as the problem domain is complex valued [3]. The complex least-squares problem associated with interferometric calibration takes the following form:

$$\min_{\check{\mathbf{g}}} \Lambda(\check{\mathbf{g}}) = \min_{\check{\mathbf{g}}} \|\check{\mathbf{r}}\|_F^2 = \min_{\check{\mathbf{g}}} \|\check{\mathbf{d}} - \check{\mathbf{v}}(\check{\mathbf{g}})\|_F^2. \quad (2)$$

where  $\Lambda$  is the objective function,  $\|\cdot\|_F$  is the Frobenius norm,  $\check{\mathbf{r}}$  (i.e.,  $\check{\mathbf{d}} - \check{\mathbf{v}}$ ) is the complex residual vector,  $\check{\mathbf{d}}$  is the complex observed visibilities vector,  $\check{\mathbf{v}}$  is the complex predicted visibilities vector, and  $\check{\mathbf{g}}$  has already been defined. Here,  $\check{\mathbf{d}}$  is obtained by chaining all  $d_{pq}$  (and their conjugates) together into a vector. The vector  $\check{\mathbf{v}}$  is formed similarly using  $g_p m_{pq} \bar{g}_q$  instead.

Manuscript received 19 February 2025.

Pieter T. Colesky and Trienko L. Grobler are with Stellenbosch University, Stellenbosch Central, Stellenbosch, Western Cape, South Africa; e-mail: 23584955@sun.ac.za, tlgrobler@sun.ac.za.

Waldo Kleynhans is with Department of Electrical, Electronic and Computer Engineering, University of Pretoria, Hatfield, Pretoria, Gauteng, South Africa; e-mail: waldo.kleynhans@up.ac.za.

Only the scalar calibration use case is considered in this paper.

### 3. Levenberg-Marquardt

Gradient descent (GD) is a method that only uses gradients to solve least-squares problems (among other problems). Gauss-Newton (GN) is a more complex approach that uses both gradients and curvature to solve least-squares problems. Levenberg-Marquardt (LM) is a method that combines GN and GD by using a damping factor  $\lambda$ . LM converges faster than GN and does not tend to overshoot or converge prematurely. This is achieved by adjusting the damping factor  $\lambda$  after each iteration. At steeper slopes,  $\lambda$  is altered to be smaller to make the update equation more like GN. When the slope is not as steep,  $\lambda$  is altered to be a bigger value to make the update equation more like GD. The LM update associated with (2) is defined as [3]:

$$\Delta\check{\mathbf{g}} = (\mathbf{J}^H \mathbf{J} + \lambda \mathbf{U})^{-1} \mathbf{J}^H \check{\mathbf{r}}, \quad (3)$$

where  $\mathbf{U} = \mathbf{I} \odot (\mathbf{J}^H \mathbf{J})$ ,  $\odot$  denotes elementwise multiplication, and  $\Delta\check{\mathbf{g}}$  is the parameter update. Furthermore, the Jacobian is defined as [3]:

$$\mathbf{J} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^H & \mathbf{A} \end{bmatrix}, \quad (4)$$

with  $\mathbf{A} = \frac{\partial \mathbf{v}}{\partial \mathbf{g}}$  and  $\mathbf{B} = \frac{\partial \mathbf{v}}{\partial \check{\mathbf{g}}}$ . Furthermore,  $\mathbf{H} = \mathbf{J}^H \mathbf{J}$  (for simplicity, we will refer to  $\mathbf{H}$  as the Hessian matrix), where the  $H$  superscript denotes the Hermitian transpose of a matrix. Using (3), the augmented antenna gain vector can be updated as follows:

$$\check{\mathbf{g}}_{k+1} = \check{\mathbf{g}}_k + \Delta\check{\mathbf{g}}_k. \quad (5)$$

### 4. StEFCal

Let  $\mathbf{G} = \text{diag}(\mathbf{g})$ ,  $[\mathbf{M}]_{pq} = m_{pq}$  and  $[\mathbf{D}]_{pq} = d_{pq}$ . For the sake of simplicity, it is assumed that the diagonal entries of  $\mathbf{M}$  and  $\mathbf{D}$  are zero. StEFCal is an algorithm that iteratively alternates between fixing and solving for  $\mathbf{G}^H$  and  $\mathbf{G}$ , respectively. The StEFCal update equation is equal to [6]:

$$g_i^{k+1} = \frac{\mathbf{D}_{:,i}^H \cdot \mathbf{Z}_{:,i}^k}{(\mathbf{Z}_{:,i}^k)^H \cdot \mathbf{Z}_{:,i}^k} = \frac{\sum_{j \neq i} g_j^k \bar{m}_{ij} d_{ij}}{\sum_{j \neq i} |g_j^k|^2 |m_{ij}|^2}. \quad (6)$$

where  $\mathbf{Z}^k = \mathbf{G}^k \mathbf{M}$ . Here,  $|\cdot|$  denotes absolute value. However, poor convergence was observed by [6] in certain scenarios, and an empirical investigation revealed that averaging every second iteration improved convergence; however, this behavior was not further analyzed or explained by the authors. This alteration dramatically improves the convergence properties and the speed of the algorithm.

## 5. Relating StEFCal and LM

In this section, we prove that the StEFCal algorithm is essentially LM with a diagonal Hessian approximation. This result was first proved in [3]. Recall that the update equation of LM is equal to:

$$\Delta\check{\mathbf{g}} = (\mathbf{J}^H \mathbf{J} + \lambda \mathbf{U})^{-1} \mathbf{J}^H \check{\mathbf{r}}. \quad (7)$$

Now consider the diagonal equivalent of  $\mathbf{H}$ , i.e.:

$$\tilde{\mathbf{H}} \approx \mathbf{I} \odot \mathbf{H}, \quad (8)$$

If we substitute (8) into (7), we obtain:

$$\Delta\check{\mathbf{g}} \approx (1 + \lambda)^{-1} \tilde{\mathbf{H}}^{-1} \mathbf{J}^H (\check{\mathbf{d}} - \check{\mathbf{v}}), \quad (9)$$

where  $\check{\mathbf{r}} = \check{\mathbf{d}} - \check{\mathbf{v}}$ . (9) can be rewritten as:

$$\Delta\check{\mathbf{g}} \approx \frac{1}{1 + \lambda} \tilde{\mathbf{H}}^{-1} \mathbf{J}^H \check{\mathbf{d}} - \frac{1}{1 + \lambda} \tilde{\mathbf{H}}^{-1} \mathbf{J}^H \check{\mathbf{v}}. \quad (10)$$

It is well known that [3]:

$$\mathbf{J}^H \check{\mathbf{v}} = \tilde{\mathbf{H}} \check{\mathbf{g}}. \quad (11)$$

Substituting (11) into (10) results in:

$$\Delta\check{\mathbf{g}} \approx \frac{1}{1 + \lambda} \tilde{\mathbf{H}}^{-1} \mathbf{J}^H \check{\mathbf{d}} - \frac{1}{1 + \lambda} \check{\mathbf{g}}, \quad (12)$$

which becomes

$$\check{\mathbf{g}}^{k+1} \approx \frac{1}{1 + \lambda} \tilde{\mathbf{H}}^{-1} \mathbf{J}^H \check{\mathbf{d}} + \frac{\lambda}{1 + \lambda} \check{\mathbf{g}}^k, \quad (13)$$

where  $1 - \frac{1}{1 + \lambda} = \frac{\lambda}{1 + \lambda}$ . If  $\lambda = 0$ , we obtain:

$$g_i^{[k+1]} = \frac{\sum_{j \neq i} g_j^k \bar{m}_{ij} d_{ij}}{\sum_{j \neq i} |g_j^k|^2 |m_{ij}|^2}, \quad (14)$$

as required. To obtain the even update equation of StEFCal, we simply need to set  $\lambda = 1$ .

## 6. Hyperparameter Optimization

Section 5 implies that StEFCal is nothing more than the use of LM with a diagonal approximation for the Hessian. As such, StEFCal essentially has two hyperparameters  $\lambda_{\text{even}}$  and  $\lambda_{\text{odd}}$ . It is common practice within the literature to set  $\lambda_{\text{even}} = 1$  (equivalent to averaging the current and previous gain estimate together) and  $\lambda_{\text{odd}} = 0$  (equivalent to using (13) as is). This begs the question: Are these hyperparameter choices optimal? Do these choices result in the fewest number of iterations needed for the algorithm to converge while successfully minimizing  $\Lambda$ ? To try and answer this

question, two empirical hyperparameter optimization approaches were employed, namely grid search (see (13) and particle swarm optimization [PSO]; see subsection 6.2). In this paper we allowed  $\lambda < 0$ .

### 6.1 Grid Search

During a grid search, different combinations of parameters defined over a given range are considered to try and find the optimal hyperparameter combination. Considering the two-dimensional case, a minimum and maximum bound for each axis is chosen first. Next, an equally spaced grid of parameters is created, which includes the grid boundaries. Each parameter combination from this grid is then tried, and the performance characteristics of the algorithm are recorded in a grid-like data structure. This data structure can be visualized as a heatmap and can be used to estimate the best hyperparameter combination.

### 6.2 Particle Swarm Optimization

PSO is a population-based stochastic algorithm, which consists of particles moving around a search space in search of the optimal point. The best global topology is used to compute and update the velocities of the aforementioned particles, which determine the direction and step size of the particle's next movement. The velocities of particles are updated during each iteration of the algorithm and are calculated using the best global and personal positions of the particles. Using the inertia weight model [9], the velocity of each particle  $i$  is updated as follows:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1r_{1ij}(t)(y_{ij}(t) - x_{ij}(t)) + c_2r_{2ij}(t)(\hat{y}_j(t) - x_{ij}(t)). \quad (15)$$

Here, the velocity and position in dimension  $j$  at time  $t$  of particle  $i$  are denoted by  $v_{ij}(t)$  and  $x_{ij}(t)$ . The hyperparameters  $c_1$  and  $c_2$  are the cognitive and social positive acceleration coefficients, and  $w$  is the inertia weight. The cognitive component contributes to particles exploiting promising areas in the search space, while the social component contributes to particles exploring fewer familiar parts of the search space. The stochastic components,  $r_{1ij}(t)$  and  $r_{2ij}(t)$ , are random scalars sampled independently from a uniform distribution  $U(0, 1)$ . The best global and personal positions are denoted with  $\hat{y}_j(t)$  and  $y_{ij}(t)$ , respectively. The following equation then updates the position of a particle:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1). \quad (16)$$

This algorithm converges when the hyperparameters adhere to the following criterion [9]:

$$c_1 + c_2 < \frac{24(1 - w^2)}{7 - 5w} \quad (17)$$

This implies that PSOs require hyperparameter tuning, which makes it challenging to apply this algorithm to the tuning of the hyperparameters of other algorithms. This problem is addressed in the following section.

### 6.3 Adaptive Particle Swarm Optimization

Hyperparameter choices for PSOs are problem dependent. Good results can be obtained if the PSO hyperparameters adhere to the following inequality [9]:

$$\frac{22 - 30w^2}{7 - 5w} < c_1 + c_2 < \frac{24 - 24w^2}{7 - 5w}. \quad (18)$$

Generally, three configurations can be used, namely, 'equal' where  $c_1 = c_2$ , 'cognitive' where  $c_1 = 3c_2$ , and 'social' where  $3c_1 = c_2$ . Self-tuning PSOs emerged next, whose hyperparameters are tuned during training. One such algorithm is the PSO with improved random constants (PSO-iRC). This PSO is used in this paper to search through the StEFCal hyperparameter search space. In the case of this algorithm, every particle is initialized with a unique set of hyperparameters that adhere to (18), while the configuration for  $c_1$  and  $c_2$  is chosen at random. A particle receives a new set of hyperparameters when its personal best position has not improved for  $k$  consecutive iterations, making the PSO self-tuning. It is worth noting that PSO-iRC is among the top-performing PSOs if it is compared with other PSO variants [9]. PSOs have been shown to be effective for hyperparameter tuning, [10] and PSO-iRC is used in this study. PSO-iRC is very effective when applied to static, single-stage optimization problems (and as such, it is highly relevant if StEFCal is to be optimized) and does not require gradients. Moreover, the PSO-iRC is easy to implement and self-tuning.

## 7. Simulator and Methodology

An existing simulator capable of generating corrupted visibilities was used in this paper [3]. The simulator works as follows. The source code is available online at: [github.com/Trienko/redundant\\_calibration](https://github.com/Trienko/redundant_calibration). A square antenna layout is generated. The number of antennas in the layout can vary. The simulator first calculates the uv-tracks of the array layout. The default observational parameter of the simulator was utilized - (monochromatic observation, decl. =  $-74^\circ 39' 37.481''$ , lat. =  $-30^\circ 43' 17.34''$ , freq. = 150 MHz, time-steps = 600, hour angle range =  $[-1^h, 1^h]$ , min. baseline length = 50 m, layout = square). A random sky model adhering to a power-law distribution is now generated (the default set of parameters for the simulator was used to generate the sky model). Antenna gains are then sampled from a uniform distribution  $U(1, u_{\max})$  with  $u_{\max} \in \mathbb{R}$ . The simulator

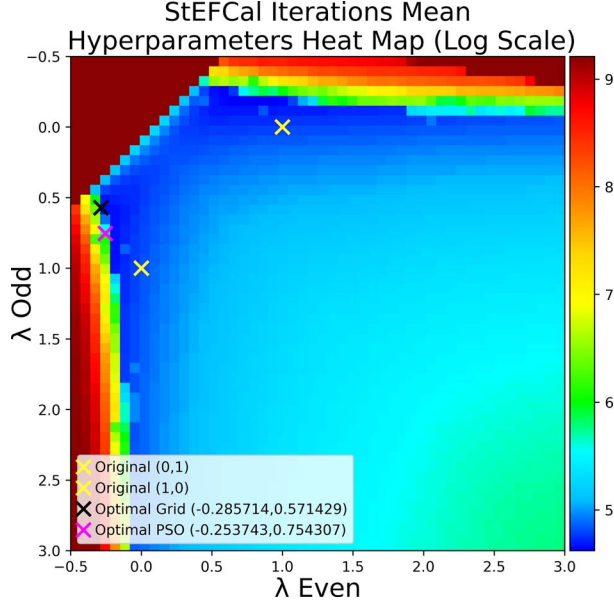


Figure 1. StEFCal average iterations to converge (log-scale) for even/odd  $\lambda$ , 4 antennas.

then generates model and corrupted visibilities. The simulator can generate visibilities with varying signal-to-noise ratios (SNRs). Here, SNR is defined as:

$$\text{SNR} = 10 \log \left( \frac{\langle \mathbf{v} \odot \bar{\mathbf{v}} \rangle_{v,t,pq}}{\langle \mathbf{n} \odot \bar{\mathbf{n}} \rangle_{v,t,pq}} \right) \quad (19)$$

where  $\langle \cdot \rangle_{v,t,pq}$  denotes averaging over frequency, time, and baseline, and  $\mathbf{n}$  is obtained by chaining together all  $n_{pq}$  into a vector. Various visibility datasets were then created or simulated using different hyperparameter combinations. The following simulator hyperparameters were used: signal-to-noise ratio (SNR) values of  $-1$ ,  $0$ ,  $10$ , and  $1000$  dB;  $u_{\max}$  values of  $2$ ,  $3$ ,  $4$ , and  $10$ ; and random seeds  $12345678$ ,  $23476934$ ,  $98976433$ , and  $49052764$ . The number of iterations it took for StEFCal and its optimized variant (optimized using a PSO) to converge (for each time-step) if applied to the aforementioned datasets was recorded (the gain vector found was also stored). The following seed values for the PSO were used:  $49052764$ ,  $12039846$ ,  $32609861$  and  $18796345$ ; value of  $k = 9$ . A maximum of  $10,000$  iterations was allowed, and the convergence error tolerance was set to  $\tau = 1 \times 10^{-5}$ . The following stopping criterion was used:

$$\tau > \|\mathbf{g}_k - \mathbf{g}_{k-1}\|_F^2 / \|\mathbf{g}_{k-1}\|_F^2. \quad (20)$$

## 8. Results

Figure 1 depicts the log of the average number of iterations StEFCal requires to converge as a function of its hyperparameters (for four antennas and a randomly chosen visibility dataset described in section 7). The currently used StEFCal hyperparameters are depicted with

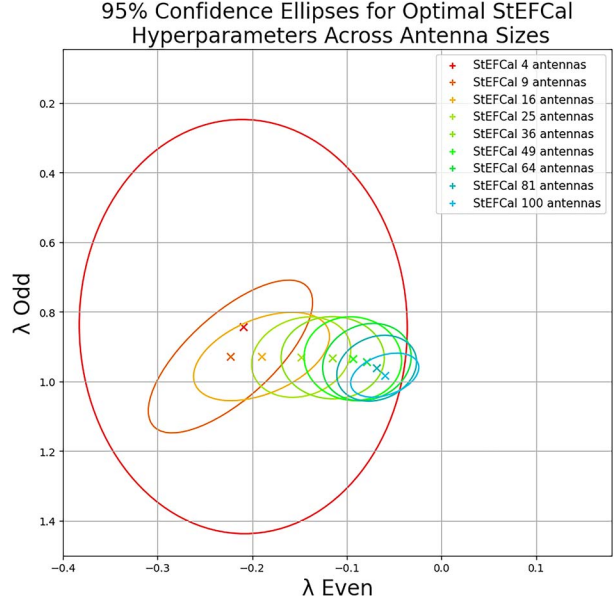


Figure 2. StEFCal optimal  $\lambda$  (even/odd): means ( $\times$ ) and 95% confidence ellipses over antenna sizes.

yellow crosses. The best hyperparameter pair (in terms of the least number of iterations), found via grid search and a PSO, is depicted in black and pink (which differs from the defaults). It appears as if the pink cross does not fall on a minimum; this is merely an artifact of the grid resolution. Figure 2 depicts the mean (represented by crosses) and the 95% confidence ellipse of the optimal StEFCal hyperparameter values that the PSO found, in general, averaged over all SNR and maximum amplitude combinations for different antenna sizes. StEFCal converged faster using these hyperparameters than it did with the default values. The mean values appear to move in the direction of  $(0, 1)$  as the number of antennas increase, which corresponds to the original StEFCal. The symmetry in the solution space was exploited to create this graph. All values in Table 1 are computed across all SNR and maximum amplitude combinations. Table 1 reports the mean and standard deviation of the number of iterations required for convergence, as well as the mean residual norm at convergence for optimal and original StEFCal. The differences between optimal and original StEFCal are also reported for each antenna size, given the aforementioned performance criteria. The results demonstrate that StEFCal with optimal hyperparameters consistently outperforms original StEFCal for all antenna sizes (except when comparing the mean residual norm in the four-antenna case; the residual difference in that case is negligible). The mean value of the hyperparameters  $\lambda$  (even/odd), which achieved optimal results for each antenna size, is provided for the reader's convenience. The convergence time was tracked, and the mean time was calculated for both optimal and original StEFCal. The row titled " $\Delta t$  (2 hours  $\times$  1 channel) [s]" reports the mean time difference for calibration, in seconds, between optimal

Table 1. Performance metrics for optimal and original StEFCal.

Number of Antennas	4	9	16	25	36	49	64	81	100
Mean iterations orig StEFCal	430.74	27.91	20.44	17.17	15.43	14.34	13.55	13	12.56
Mean iterations opt StEFCal	399.96	23.72	17.44	15.04	13.77	12.95	12.28	11.8	11.42
Mean iterations difference	+30.79	+4.2	+3.0	+2.14	+1.66	+1.4	+1.27	+1.2	+1.14
Improved mean iterations, %	7.15	15.04	14.70	12.45	10.76	9.7	9.38	9.22	9.11
Standard deviation iterations orig StEFCal	1782.79	13.2	5	2.88	2.24	1.97	1.72	1.56	1.43
Standard deviation iterations opt StEFCal	1733.97	11.21	3.97	2.2	1.72	1.58	1.42	1.36	1.29
Standard deviation iterations Difference	+48.81	+1.98	+1.01	+0.68	+0.52	+0.4	+0.3	+0.2	+0.13
Mean residual norms orig StEFCal	319.86	1059.07	2009.05	3253.66	4806.32	6592.74	8661.17	11022.58	13633.96
Mean residual norms opt StEFCal	320.63	1059.07	2009.04	3253.65	4806.3	6592.72	8661.14	11022.56	13633.94
Mean residual norms difference	-0.77	0	0	+0.01	+0.02	+0.02	+0.02	+0.02	+0.02
$\lambda$ even mean	-0.2095	-0.2231	-0.1902	-0.1483	-0.1150	-0.0937	-0.0788	-0.0685	-0.0599
$\lambda$ odd mean	0.8423	0.9282	0.9281	0.9301	0.9317	0.9344	0.9442	0.9619	0.9819
$\Delta t$ (2 hours $\times$ 1 channel) [s]	0.0402	0.0175	0.0247	0.0309	0.0364	0.0440	0.0553	0.0780	0.0818
$\Delta t$ (10000 hours $\times$ 1000 channels) [h]	55.77	24.26	34.33	42.85	50.61	61.11	76.81	108.31	113.62

For StEFCal optimal: green = better, yellow = no difference, red = worse.

and original StEFCal for a 2-hour observational period for a single channel. An Acer Predator PT314-51s was utilized for the experiments. Important specs are: CPU = 11th Gen Intel<sup>®</sup> Core<sup>™</sup> i7-11370H; RAM = 24 GiB. A scaled scenario is depicted in the row titled “ $\Delta t$  (10000 hours  $\times$  1000 channels),” representing the calibration time difference in hours, more than 10,000 observational hours for 1000 channels.

## 9. Conclusion

Figure 2 and Table 1 suggest that optimal StEFCal tends to the original StEFCal (both in terms of performance and optimal  $\lambda$  values) as the number of antennas increases. Table 1 highlights the fact that if a PSO is used to find optimal hyperparameters for StEFCal, then StEFCal can converge faster than conventional StEFCal does. The row “ $\Delta t$  (2 hours  $\times$  1 channel) [s]” in Table 1 suggests a small improvement in general and may seem insignificant at first glance. In a real-world scenario, more observational hours and channels are used. Therefore, the time difference when calibrating 10,000 observational hours for 1000 channels becomes significant. Leaving StEFCal unoptimized, the compounded time wasted over many observational hours results in a considerable amount of wasted computing.

## 10. References

1. A. R. Thompson, J. M. Moran, and G. W. Swenson Jr., *Interferometry and Synthesis in Radio Astronomy*, Heidelberg, Germany, Springer Nature, 2017.
2. A. R. Thompson and L. R. D’Addario, “Frequency Response of a Synthesis Array: Performance Limitations and Design Tolerances,” *Radio Science*, **17**, 2, 1982 pp. 357-369.
3. O. Smirnov and C. Tasse, “Radio Interferometric Gain Calibration as a Complex Optimization Problem,” *Monthly Notices of the Royal Astronomical Society*, **449**, 1, February 2015, pp. 1-15.
4. U. M. Sob, H. L. Bester, O. M. Smirnov, J. S. Kenyon, and C. Russeeawon, “Solution Intervals Considered Harmful: On the Optimality of Radio Interferometric Gain Solutions,” *Monthly Notices of the Royal Astronomical Society*, **504**, 2, April 2021, pp. 1714-1732.
5. S. Yatawatta and I. M. Avruch, “Deep Reinforcement Learning for Smart Calibration of Radio Telescopes,” *Monthly Notices of the Royal Astronomical Society*, **505**, 2, May 2021, pp. 2141-2150.
6. S. Salvini and S. Wijnholds, “Fast Gain Calibration in Radio Astronomy Using Alternating Direction Implicit Methods: Analysis and Applications,” *Astronomy & Astrophysics*, **571**, October 2014, p. A97.
7. J. Kenyon, *CubiCal: a fast radio interferometric calibration suite exploiting complex optimisation*, Ph.D. dissertation, Rhodes University, Makhanda, South Africa, 2019.
8. O. M. Smirnov, “Revisiting the Radio Interferometer Measurement Equation: I. A Full-Sky Jones Formalism,” *Astronomy & Astrophysics*, **527**, February 2011, p. A106.
9. K. R. Harrison, A. P. Engelbrecht, and B. M. Ombuki-Berman, “An Adaptive Particle Swarm Optimization Algorithm Based on Optimal Parameter Regions,” presented in the IEEE Symposium Series on Computational Intelligence, Honolulu, HI, November 27–December 1, 2017.
10. P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, “Particle Swarm Optimization for Hyper-Parameter Selection in Deep Neural Networks,” Proceedings of the Genetic and Evolutionary Computation Conference (GECCO ’17), Berlin, Germany, July 15–19, 2017, pp. 481-488.