Particle Swarm Optimisation in GW signal Detection

Rajesh Kumble Nayak
IISER Kolkata

With Souradeep Pal

Rajesh Nayak
Sukanta Bose

Varun Srivastava
Souradeep Pal
Anurada Samajdar

Shubhagata Bhaumik
Ankit Mandal
Tathagata Pal
Aritra Aich

Gopi Patel
Shobhit Ranjan
Bhuvanambiga Pari
Debasmita Nandi

AAPCOS-2023 @SINP

IISER KOLKATA

cessi

LIGO–India Scientific Collaboration

LSC

Outline

- GW Detection Problem
- Optimisation Detection Statistics
- Particle Swarm Optimisation(PSO)
- Usefulness of PSO in GW detection problem
- Applying PSO based application on real data
- Results and Conclusions

GW Detection Problem

The output from a GW detector, a time series $s(t)$ can be written as:

$$s(t) = \begin{cases} n(t) & \text{in absence of GW signal} \\ n(t) + h(t) & \text{in presence of GW signal} \end{cases}$$

Here, n(t) is detector noise and h(t) is GW signal from astrophysical sources.

Signal is astrophysical modelled based on Einstein or alternative theory which we call a signal template, $q(t;\theta)$

We expect that the model signal/template match with astrophysical GW signal for specific model parameter say θ_0.

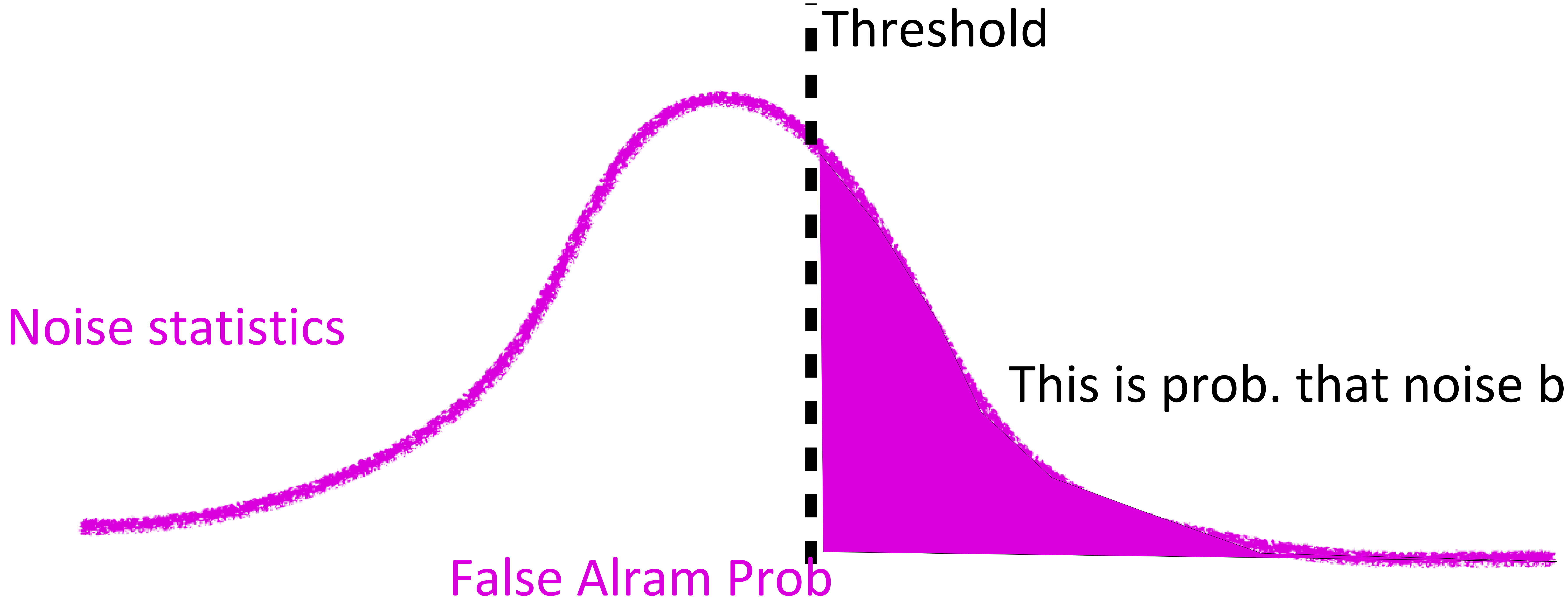
Our goal is to pick the part of detector output with astrophysical signal

Neyman–Pearson criterion

The detection probability is given by the match or weighted correlation with noise PSD, $S(f)$, is given by,

$$R(\theta) = 4\int_{f_{min}}^{f_{max}} \frac{\tilde{s}(f)\ \cdot \tilde{q}(f;\theta)}{S(f)} df$$

It can be shown that $R(\theta)$ is optimal statistic for Guassian noise.

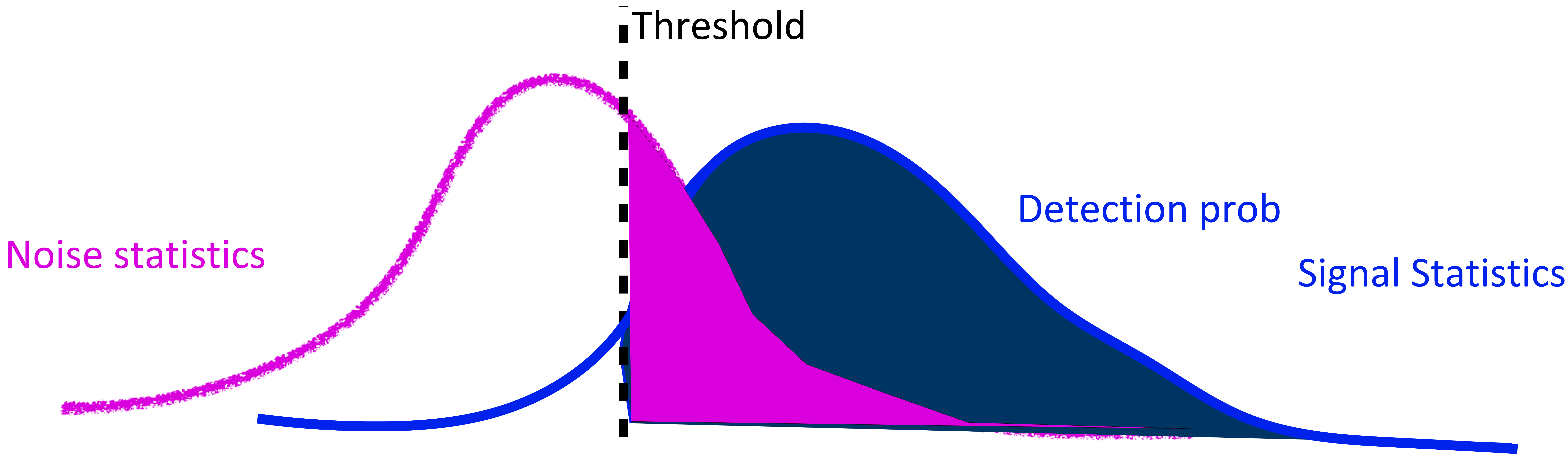


Neyman–Pearson criterion

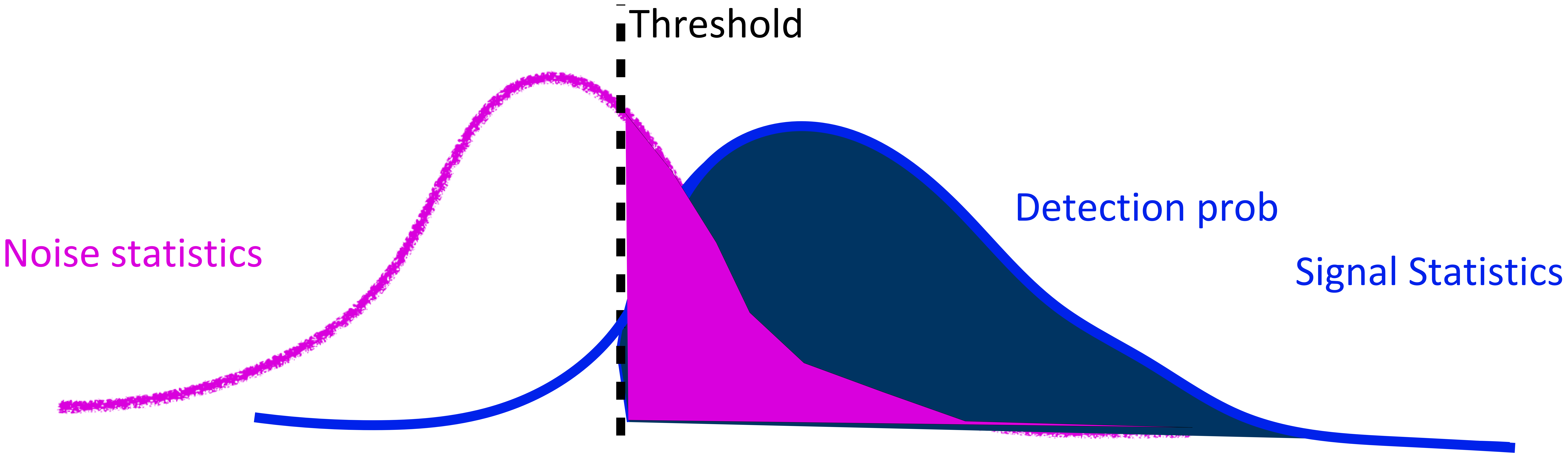
The detection probability is given by the match or weighted correlation with noise PSD, $S(f)$, is given by,

$$R(\theta) = 4\int_{f_{min}}^{f_{max}} \frac{\tilde{s}(f) \cdot \tilde{q}(f;\theta)}{S(f)} df$$

It can be shown that $R(\theta)$ is optimal statistic for Guassian noise.



Neyman−Pearson criterion



Neyman−Pearson criterion: Maximise the detection statistic over model parameters θ using a threshold provided by a given false-alarm probability.

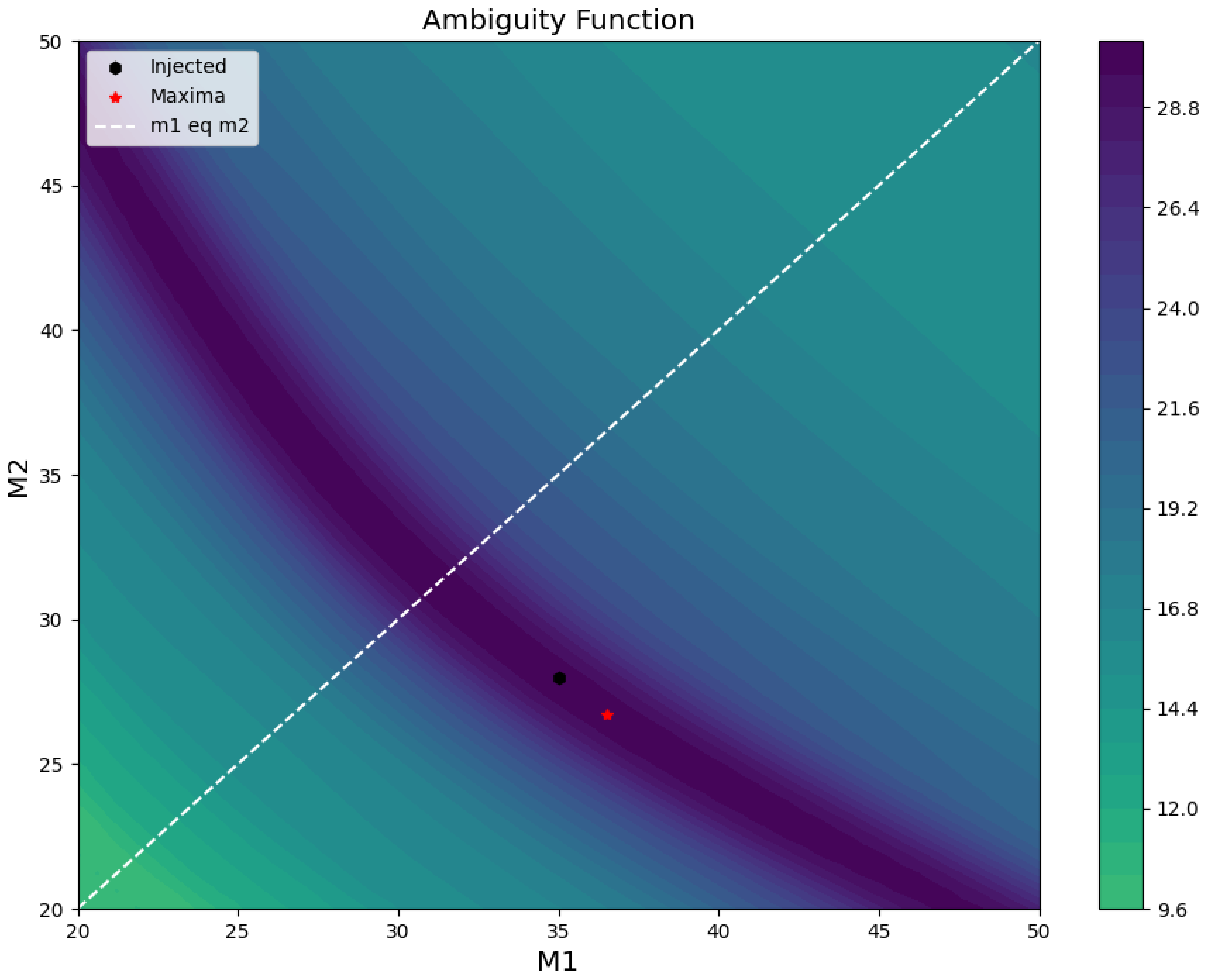
Some interesting points!

Match-making is expensive!

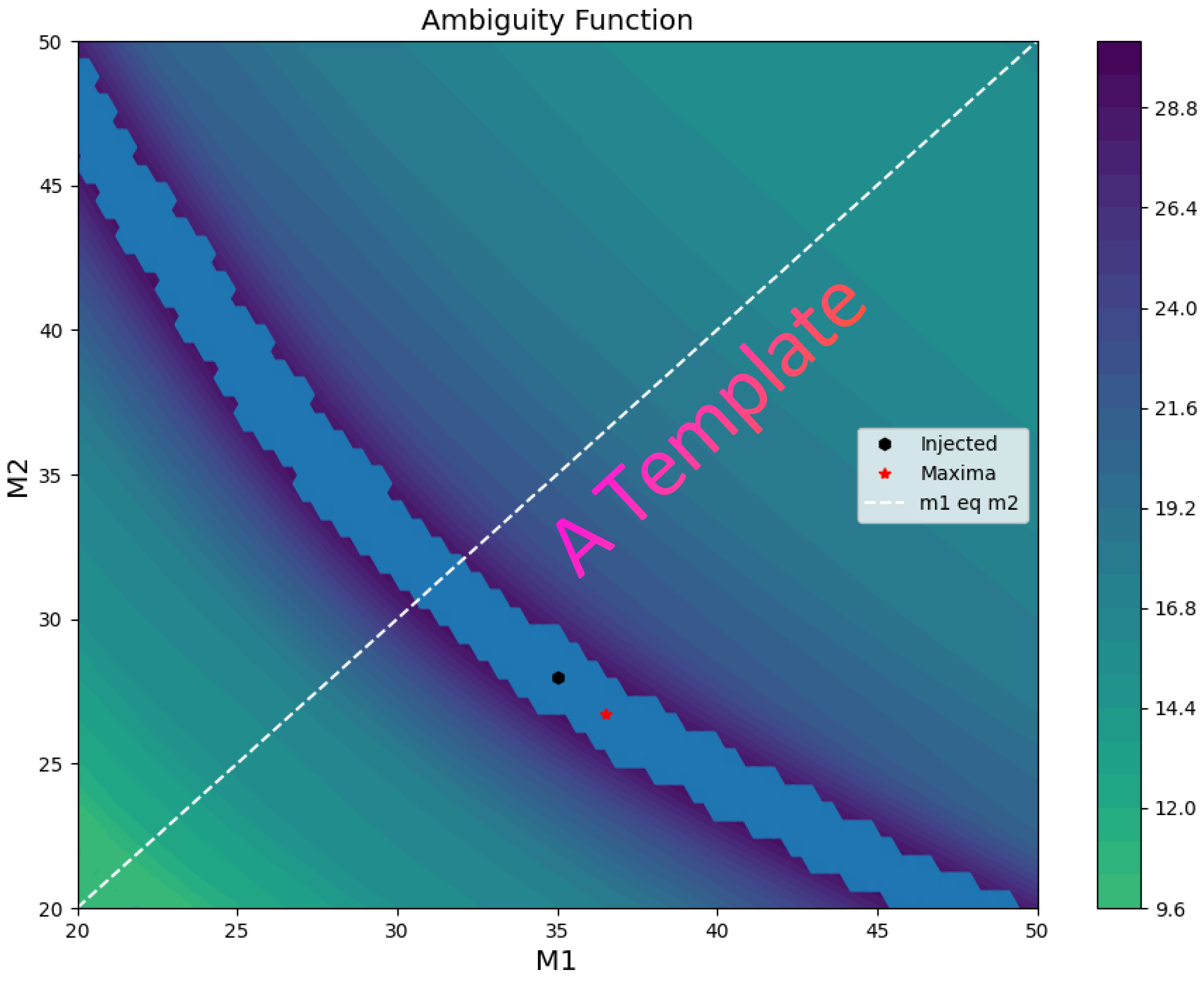
$$R(\theta) = 4\int_{f_{min}}^{f_{max}} \frac{\tilde{s}(f)\ \cdot \tilde{q}(f;\theta)}{S(f)} df$$

- Even for the simple waveforms.
- Complex waveforms contribute additional difficulty

Noise brings in local peaks and maximisation scheme can get trapped



Some interesting points!



Particle Swarm optimisation!

In this method, a set of particles makes a “controlled” random walk in given parameter space to optimise a given function. Members share helpful information with the entire swarm and converge to an optimal point in the parameter space.

We start with an uniform distribution of particles in a N $-$dimensional parameters with position of a particle at $n-th$ step is given by $\vec{X}_n$

The velocity of $\vec{V}_n$ at $n-th$ step evolves as per rule

$$\vec{V}_{n+1} = \alpha\, r_0\, \vec{V}_n + \beta\, r_1 \left(\vec{X}_{pbest} - \vec{X}_n\right) + \gamma\, r_2 \left(\vec{X}_{gbest} - \vec{X}_n\right)$$

$\alpha, \beta \text{and} \gamma$ are parameters of the algorithm, (r_0, r_1, r_2) are set of random number between $(0,1)$.

$\vec{X}_{pbest}$ is the best location sampled by a given particle converge to an optimal point in the parameter space.

Particle Swarm optimisation!

We start with a uniform distribution of particles in a N −dimensional parameters with position of a particle at $n-th$ step is given by $\vec{X}_n$

The velocity of $\vec{V}_n$ at $n-th$ step evolves as per rule

$$\vec{V}_{n+1} = \alpha\, r_0\, \vec{V}_n + \beta\, r_1 \left(\vec{X}_{pbest} - \vec{X}_n\right) + \gamma\, r_2 \left(\vec{X}_{gbest} - \vec{X}_n\right)$$

α, βandγ are parameters of the algorithm, (r_0, r_1, r_2) are set of random number between (

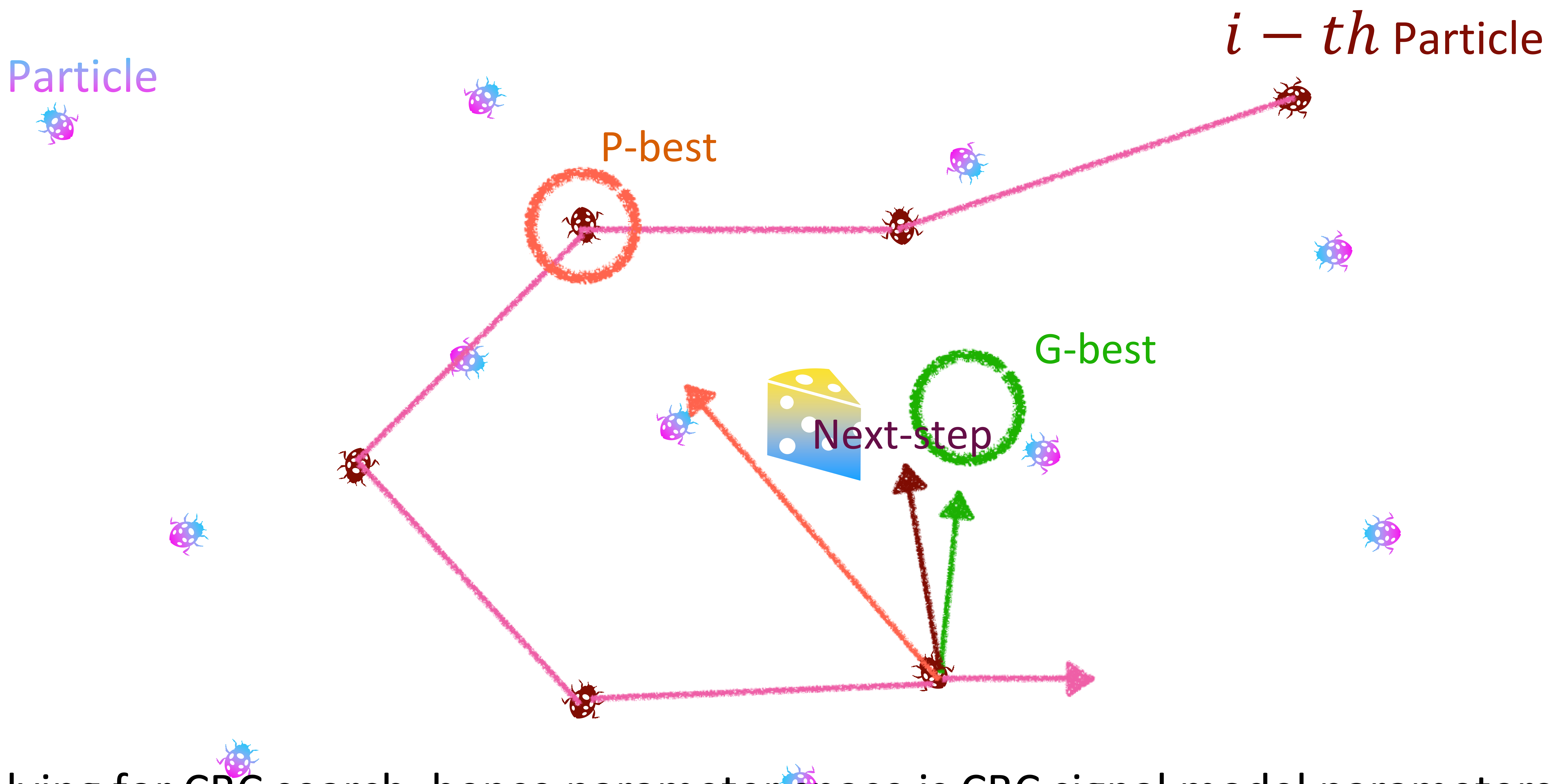
$\vec{X}_{pbest}$ is the best location sampled by a given particle till current step

$\vec{X}_{gbest}$ is the best location sampled by entire swarm

The position evolution at $n-th$ step is given by

$$\vec{X}_{n+1} = \vec{X}_n + \vec{V}_{n+1}$$

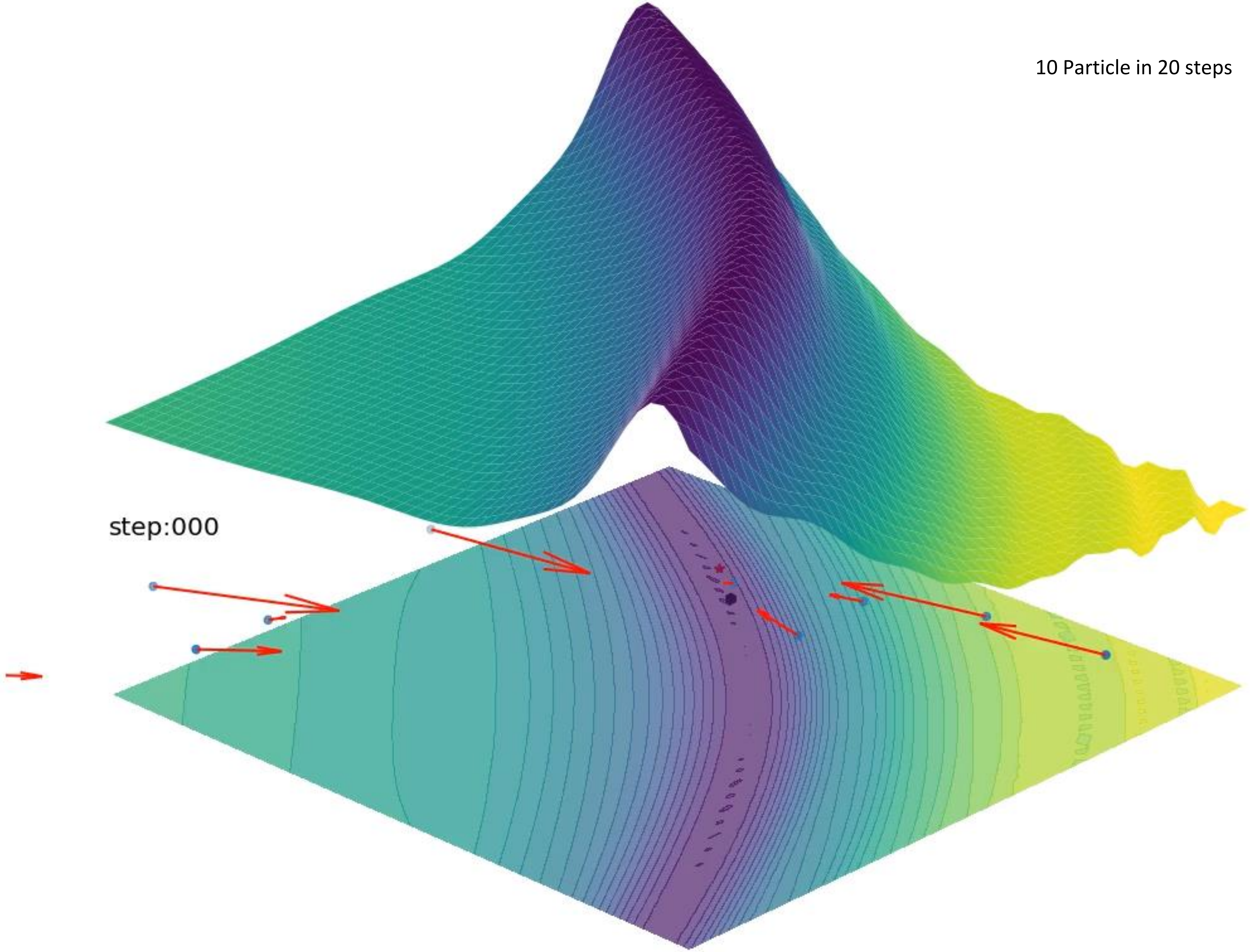
Particle Swarm optimisation!



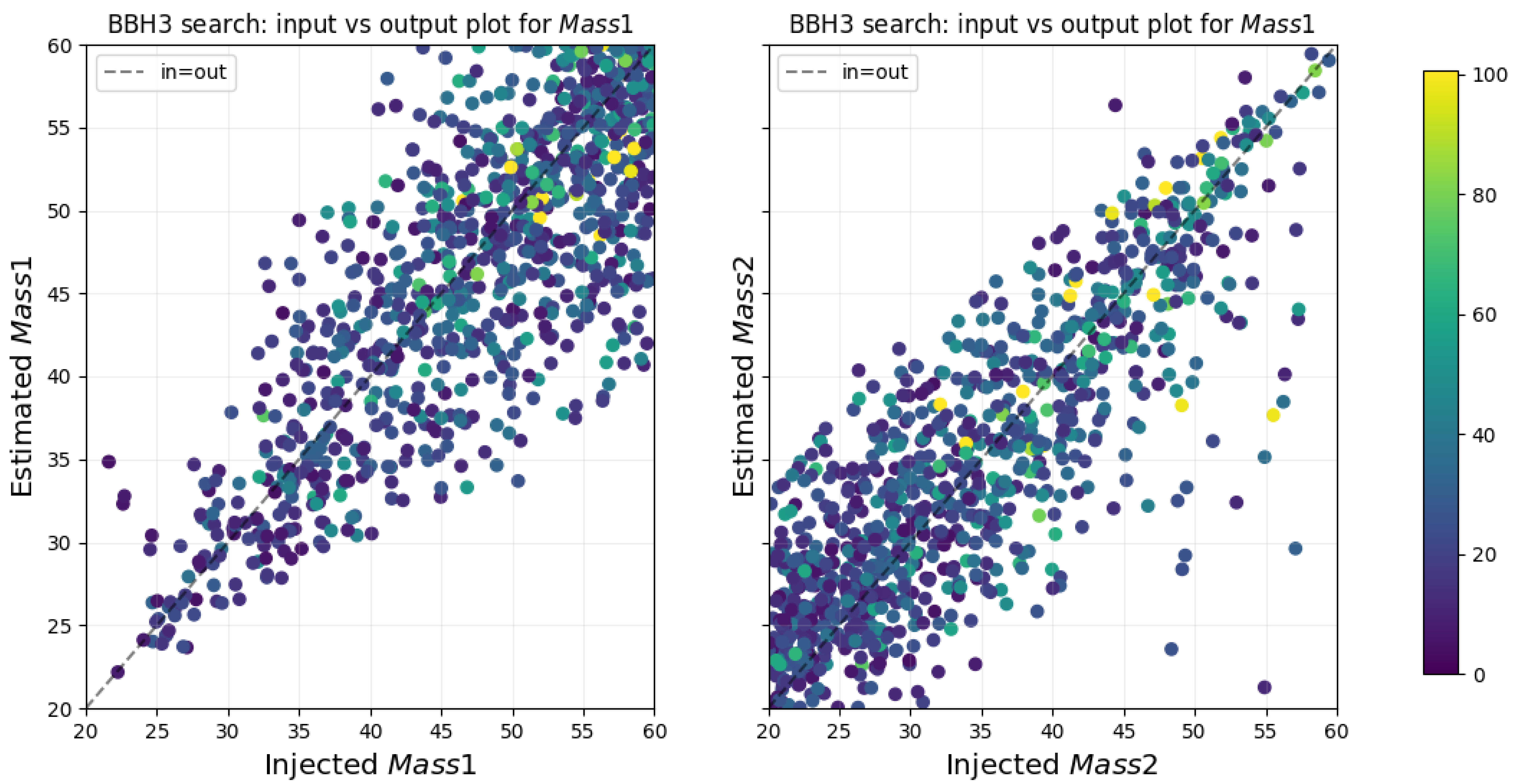
We are applying for CBC search, hence parameter space is CBC signal model parameters,

while optimising the statistics $\mathcal{L}(\Theta) = \int_{f_{min}}^{f_{max}} \tilde{s}(f) \frac{\tilde{h}(f;\Theta)}{S_h(f)} df$

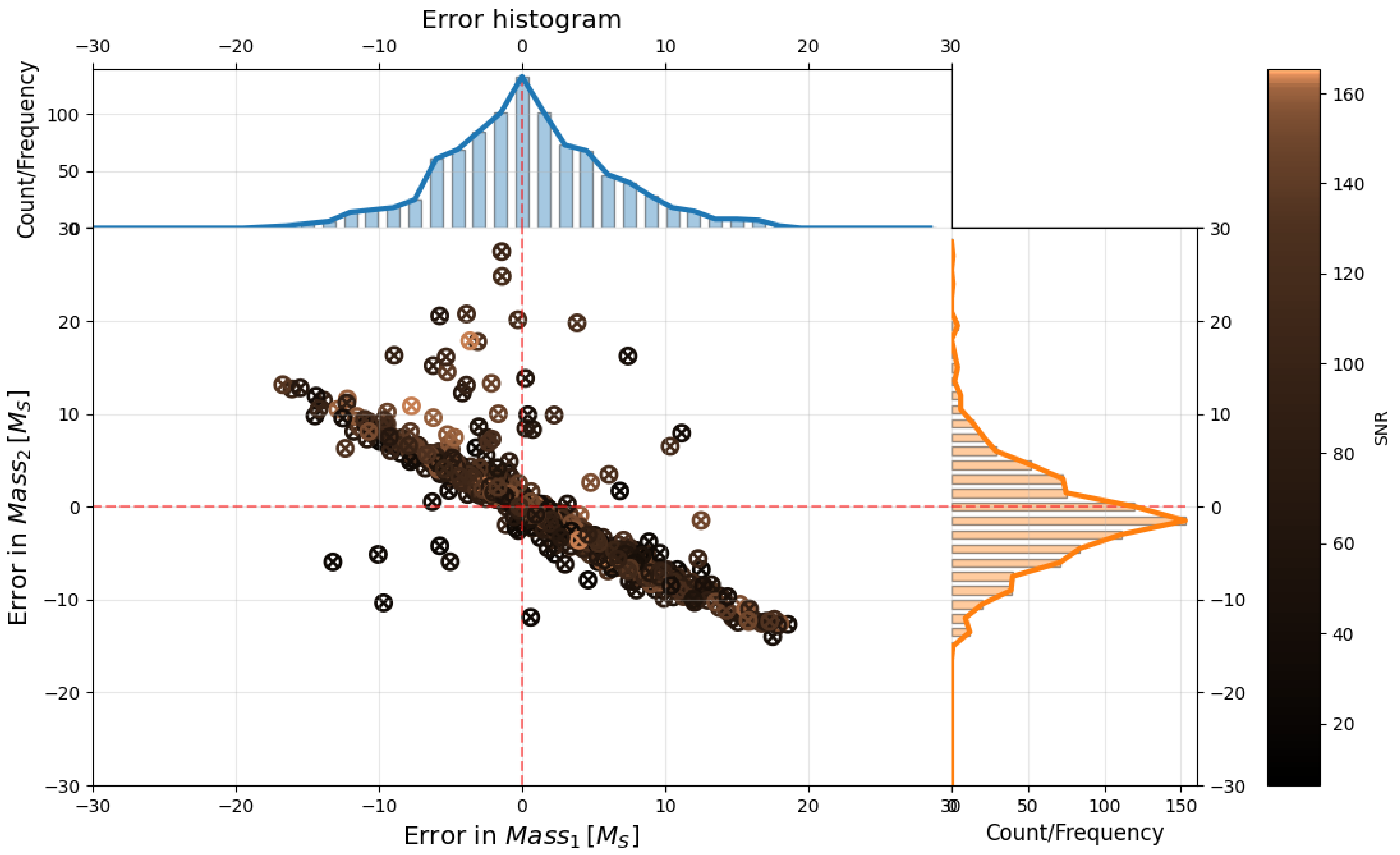
10 Particle in 20 steps



Injected Vs Estimated M_1 and M_2 Non-spinning



Error Estimated M_1 and M_2 aligned-spin



Real Detector data

Real detector data is far more complex.

Noise glitch

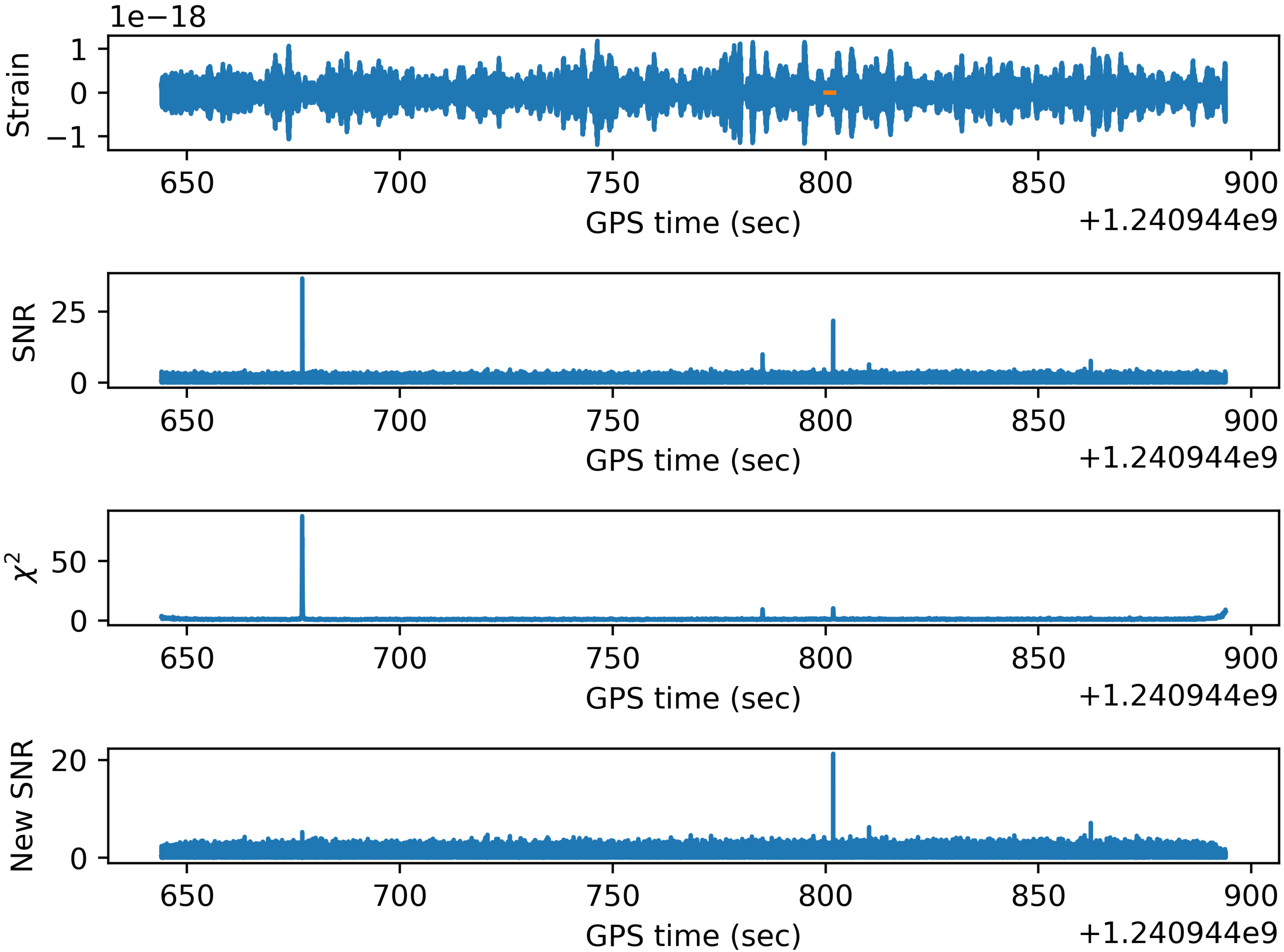
Unlike simulated noise, real noise generated lots of high SNR triggers

Here we need χ^2 for discriminating noise from signal and we use new-SNR.

However, for reducing the computational cost, we compute new-SNR only if SNR cross a thresho

As PSO evolves in the parameters space while optmising SNR, whenever the SNR cross a certain

Remaining trigger's are stored



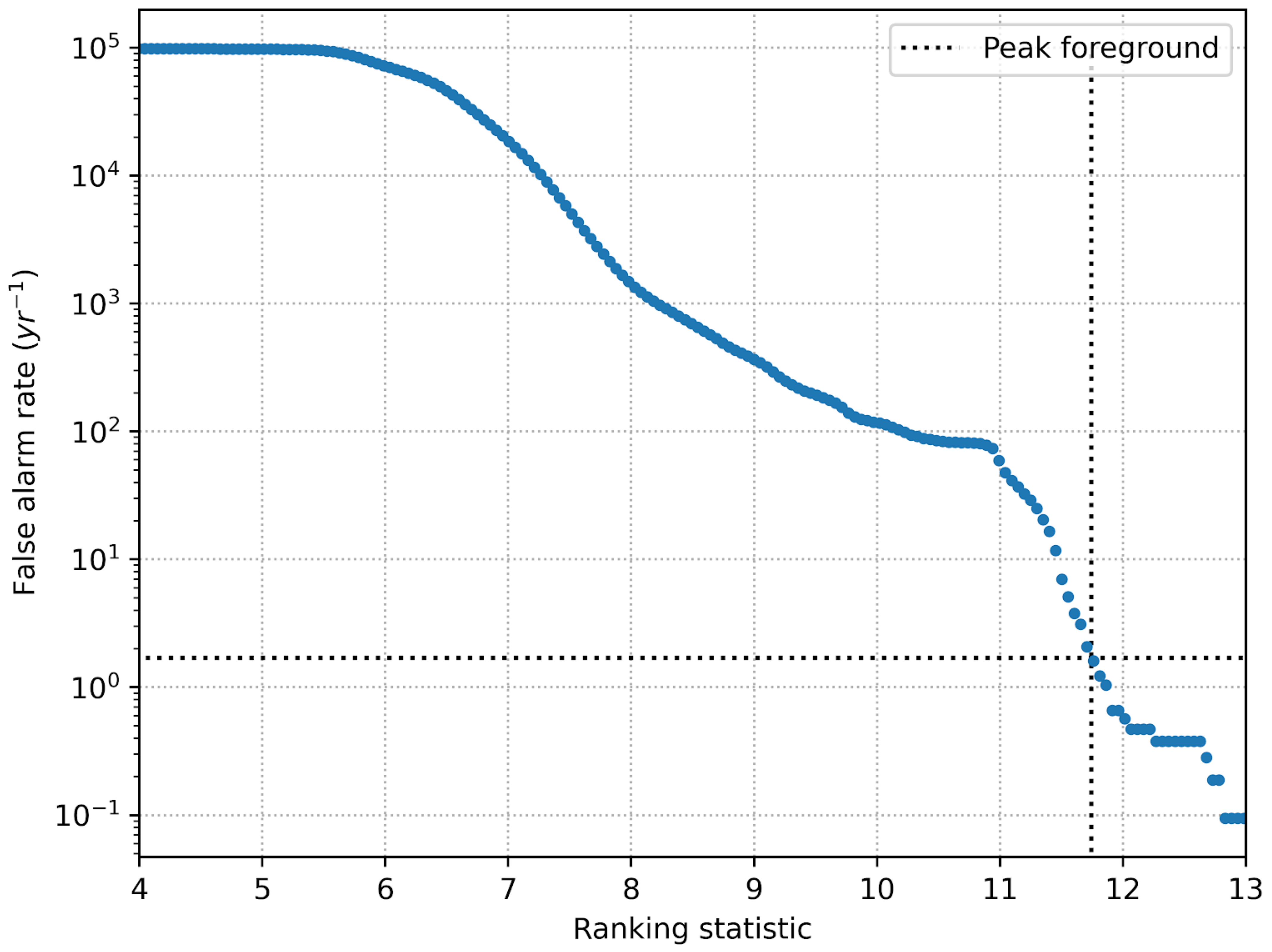
1e−18
Strain
1
0
−1
SNR
25
0
χ²
50
0
New SNR
20
0
650
700
750
800
850
900
GPS time (sec)
+1.240944e9

Significance estimation - Coincidence mode!

- We use standard time-sliding mechanism to compute the false alarm.
- All the trigger (with new-SNR) are collected from each IFO
- Trigger are grouped (with arrival time) to generate a event. Simple time-clustering and averaging is currently employed. This may be developed further in future.
- Triggers forming a coincidence give foreground candidates that may be further constrained by network SNR (somewhere 8-9).
- Triggers shifted in time by more than time-of-flight (~ 10 ms for HL) plus a timing error (some 5ms) between two detectors give rise to background events.

Example: GW190503_185404

- We use only H1, L1 data (no trigger in V1).
- Analysis duration ~ 4096 sec around each event.
- Time-shift interval ~ 50 ms.
- Background time generated ~ 10 yr [= 4096 sec x 4096 sec) / (50 ms)]
- FAR (of a foreground event) = (No. of background events louder than the given foreground event) / (Total background time generated).
- If the (no. of background events louder than the given foreground event) < 1, then assign a FAR: < 1 / (Total background time generated).



Concluding remarks

- PSO based search pipe-line is implemented and successfully applied on real data. Not need for a prior template bank.
- Useful when when it becomes hard or impossible to compute template bank.
- Better source parameter are by-product of search pipe-line, no need for additional rapid PE
- Extremely useful if search parameter have to extended.
- Fully precessing search [Varun Srivastava, K Rajesh Nayak, Sukanta Bose: arXiv:1811.02401]
- Eccentric search [LIGO-G2200981]

Acknowledgements

- We have extensively used the various tools provided by the PyCBC package
- We have used O3 data provided by the GWOSC.
- We acknowledge Sarathi cluster @ IUCAA
- We acknowledge funding by MHRD under FAST scheme for CESSI