Noisy kriging-based optimization with online resource allocation

V. Picheny, Ecole Centrale de Paris

D. Ginsbourger, Université de Berne

Y. Richet, IRSN

Optimization of simulators with tunable fidelity

- Two examples:
  - Partially converged simulations
  - Monte-Carlo simulators

- Each observation is a trade-off between rapidity and accuracy

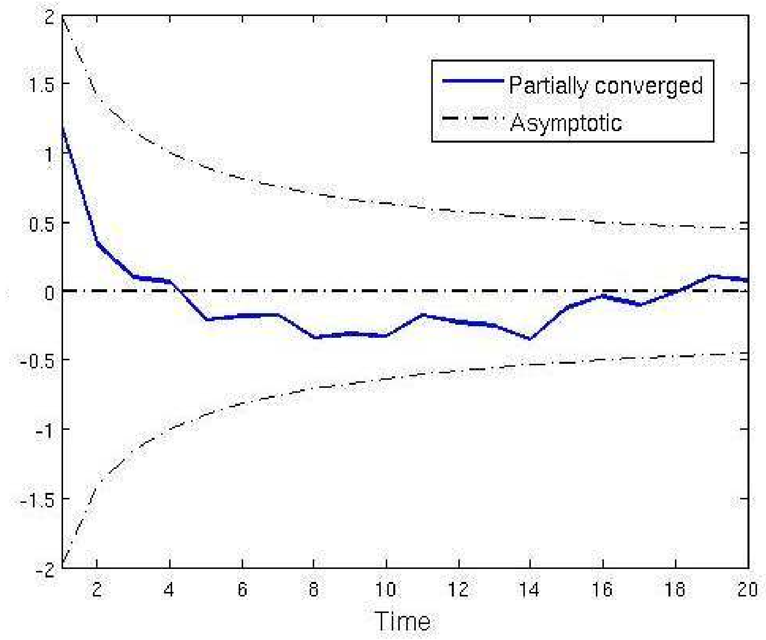
- Objectives:
  - Use it as an additional degree of freedom
  - Optimizing with limited computational resource

Assumptions

- Random noise, no bias
- Noise variance decreases with computational time
- The Monte-Carlo case:

$$\begin{cases} y_i = y(x_i) + \varepsilon_i \\ \varepsilon_i \sim N\left(0, \tau^2(x_i, t_i)\right) \\ \tau^2(x,t) = \dfrac{f(x)}{t} \end{cases}$$

- Response convergence is tractable *on-line*



Key concepts and objectives

- On-line allocation
  - Allocate computational time adapted to each design
  - Detect when adding computational time will not provide valuable information
  - Allows early stop / accurate simulations

- Finite time strategy
  - Computational time is limited by resources and simulator complexity
  - Our trade-off is necessarily driven by this limitation

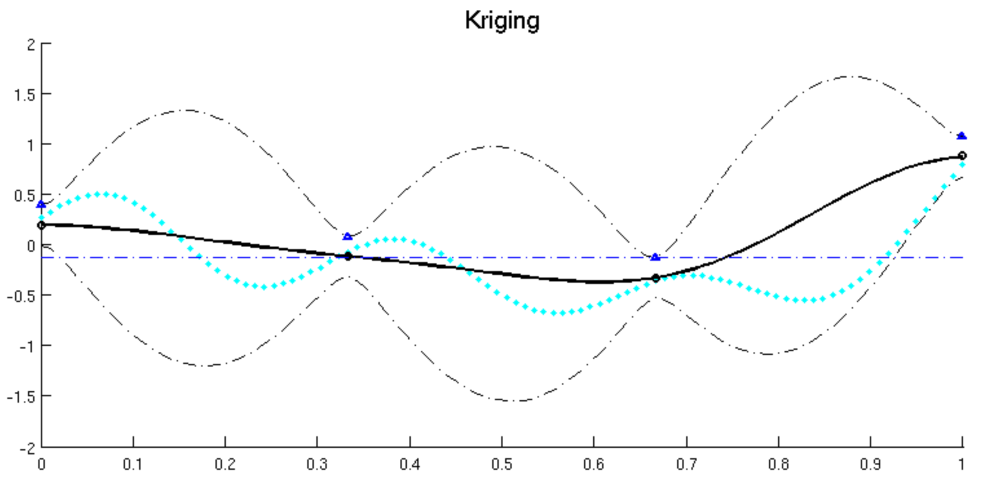
The quantile-based EI

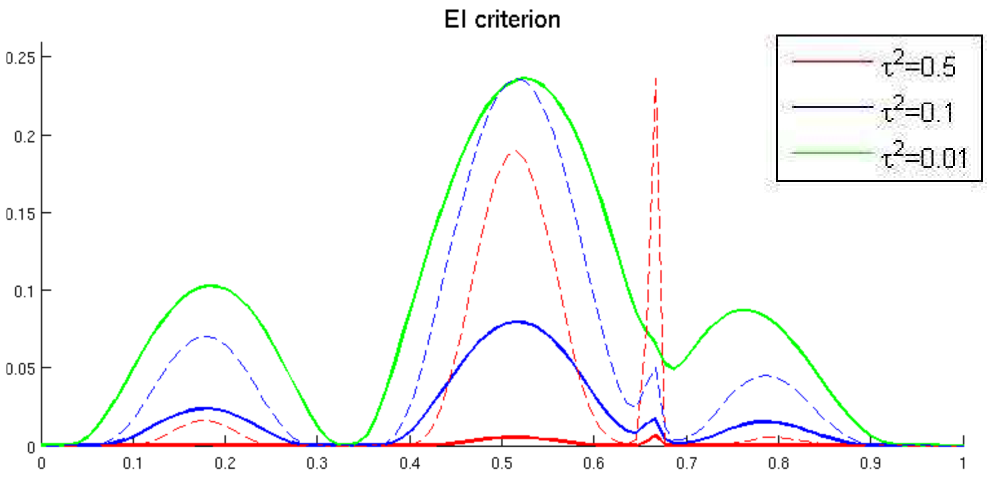
- We defined a criterion that allows us to:
  - Choose the best experiment for a given future noise level
  - Decide after the optimization which design is best

- The EI can be updated *on-line*

- Open question: choice of the future noise

Influence of future noise level





- Criterion computed for several noise levels of the new observation
- With small noise: equal to classical EI
- With large noise:
  - New observation does not change the Kriging
  - EI is maximum at data points

Choice of the noise level for on-line allocation

- Natural idea: evaluate the interest of a **single time step**

  ⟹ EI would show by how much we expect to decrease the quantile with one time step

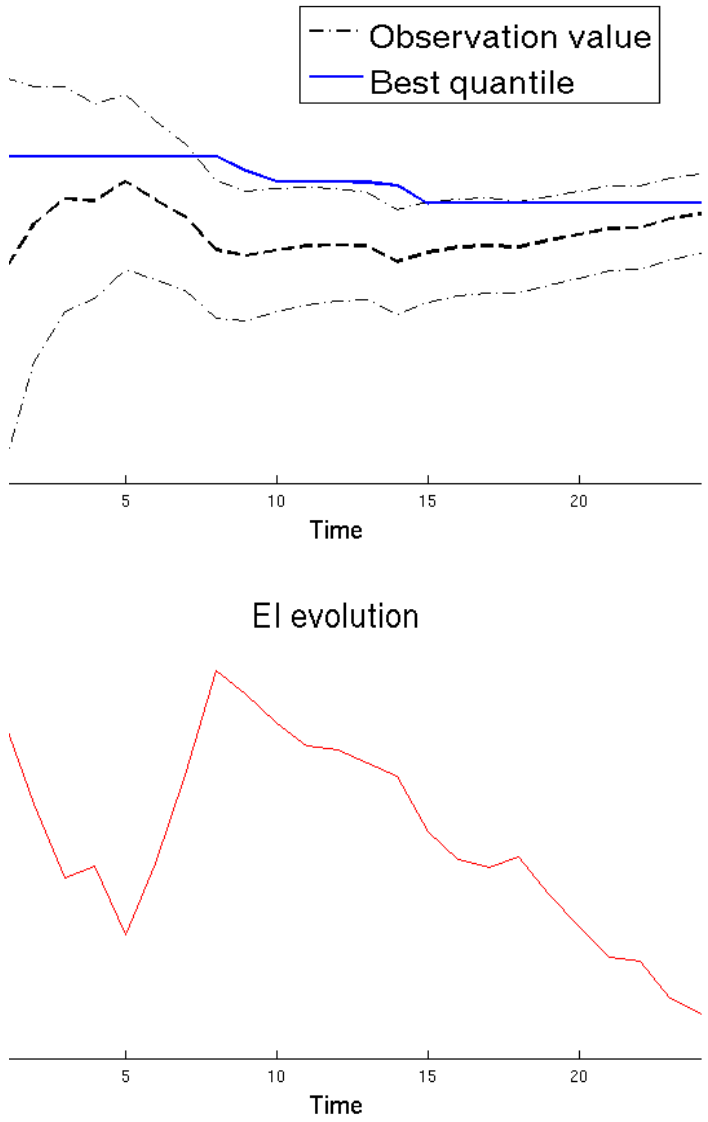
- Problem: EI would be $\approx$ zero everywhere

- Proposition: use the value of the **smallest noise achievable**
  - Noise can be bounded by the user (solver tolerance)
  - Noise is always bounded by the computational resource

  ⟹ EI shows the ultimate gain achievable by this observation

Illustration

- EI measures by how much we can improve our decision
- It can be re-evaluated for each time step at the current design
  - EI decreases when observation becomes accurate
  - If the design is 'better than expected': EI increases
  - If the design is 'worse than expected': EI decreases faster



Consequences

- The ‘smallest noise achievable’
  - depends on the computational resource
  - increases during the optimization
- The algorithm behaves differently at the beginning and the end of the optimization !
  - Beginning: enhances exploration
  - End: avoids visiting new sites
- **The strategy takes into account the limited computational resource**

Algorithm overview

Initialization
- Define computational budget T
- Generate initial DoE
- Build metamodel

While $T > 0$

Select experiment
Choose new design that maximizes EI(T)

On-line allocation
While $EI > EI_{init}/2$
- Add one time step, update observation
- Update metamodel
- Update $T = T - t_{step}$
- Update EI

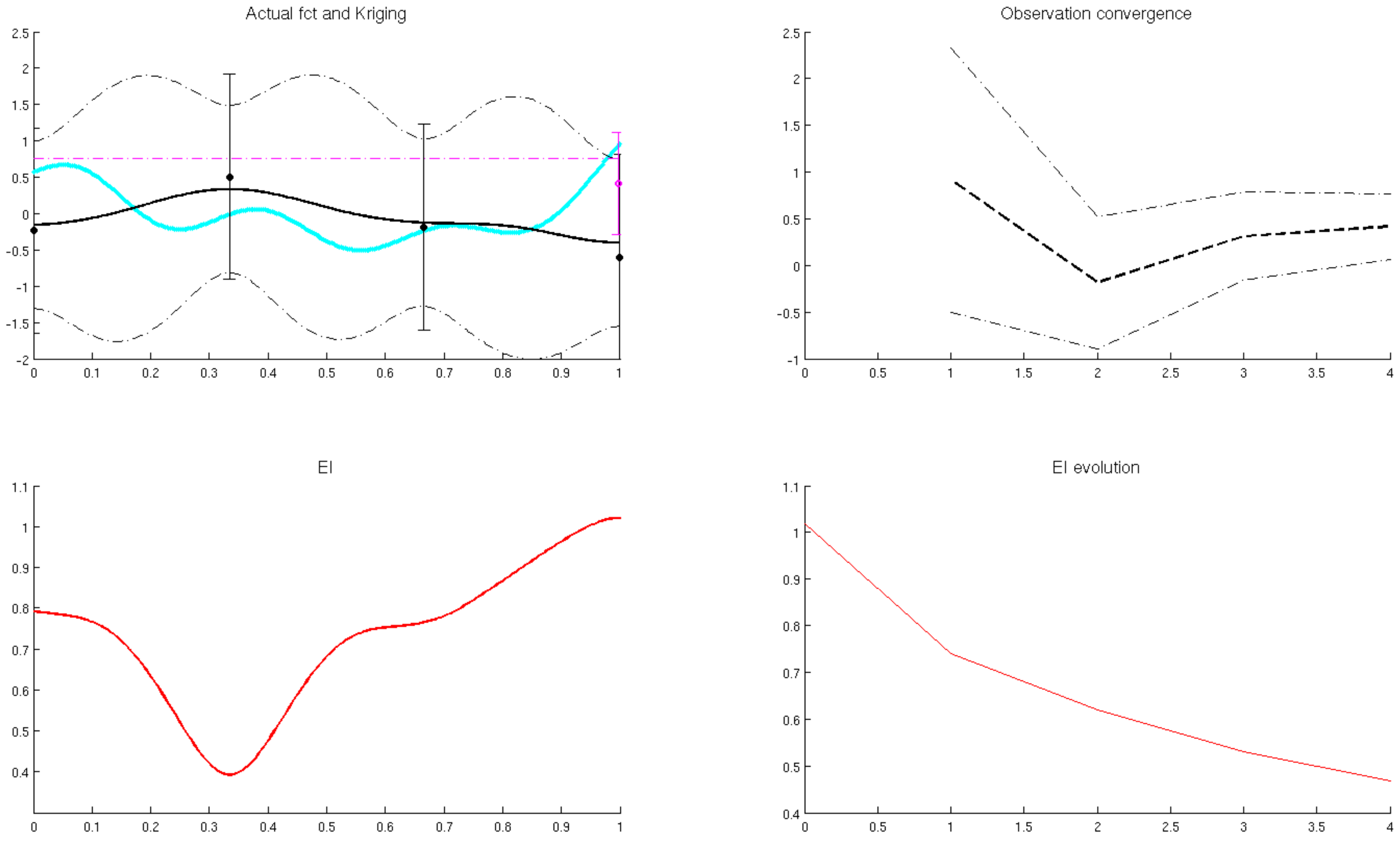
Choose final design based on Kriging quantile

Example

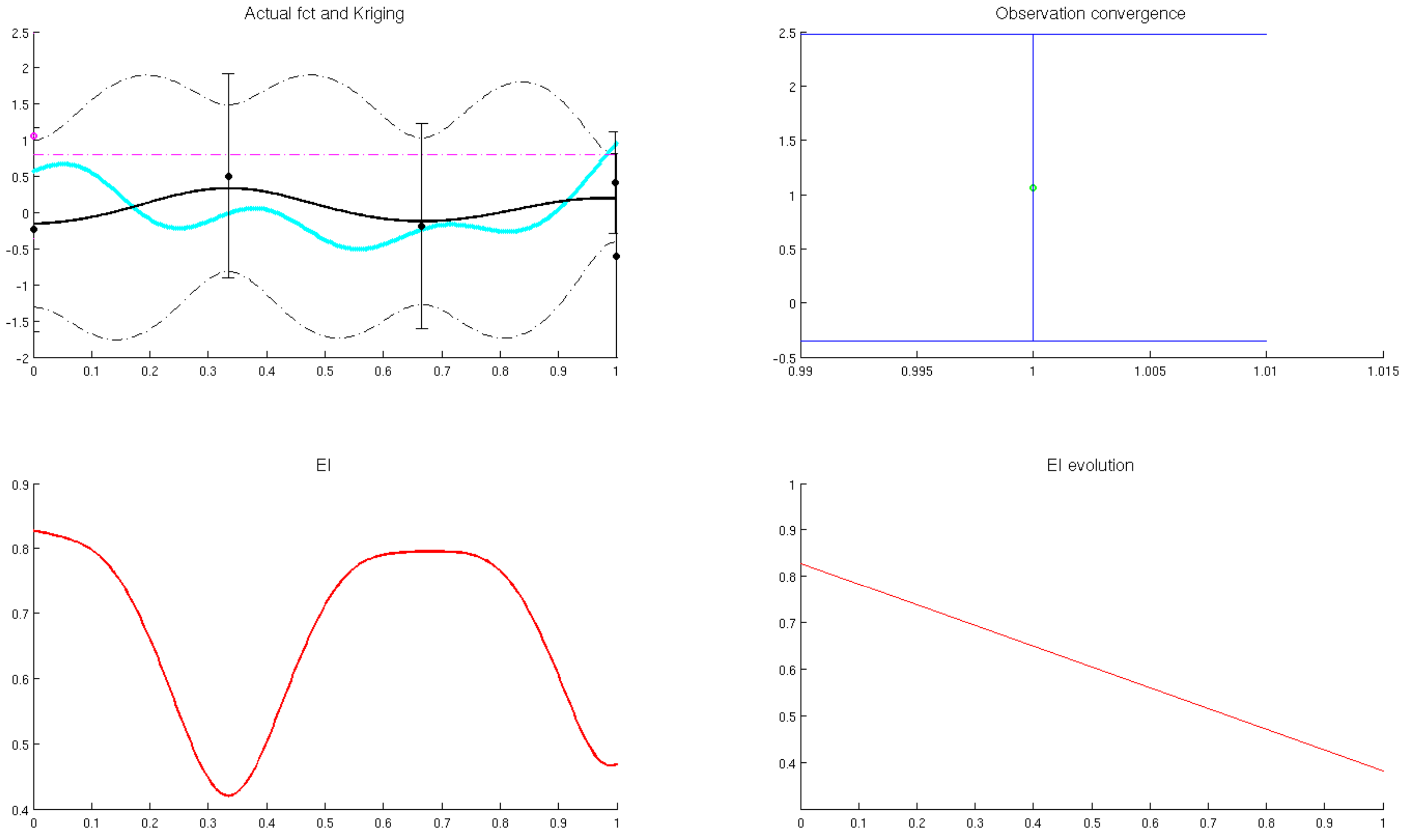
- 1D function
- Normally distributed error
- $\mathrm{var}(\varepsilon) = 0.5 / t$
- Total time $T = 100$
- Time is divided in 100 increments

- We distinguish here:
  - Algorithm iterations
  - Time steps

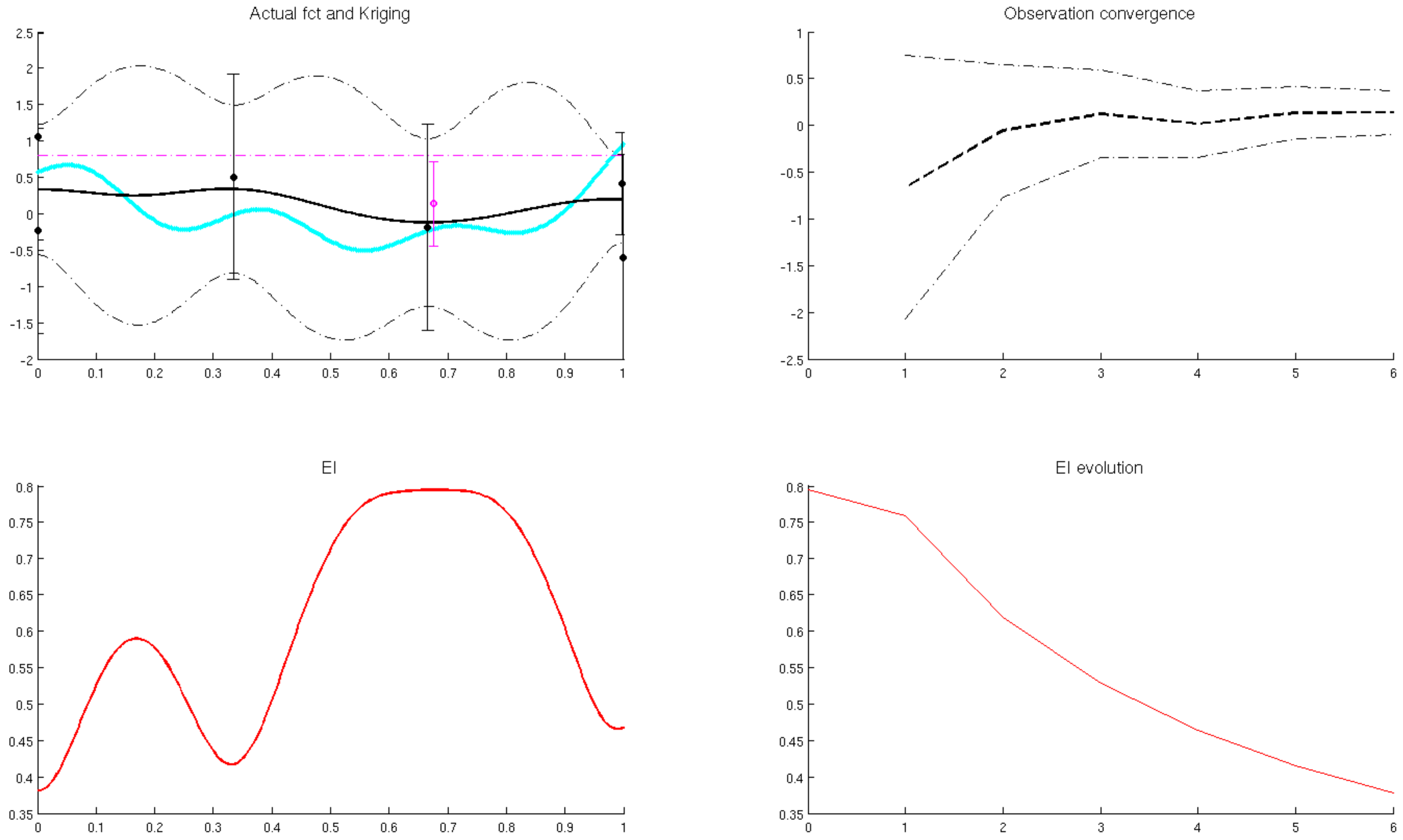
Iteration 1: 4 steps used / 92 remaining



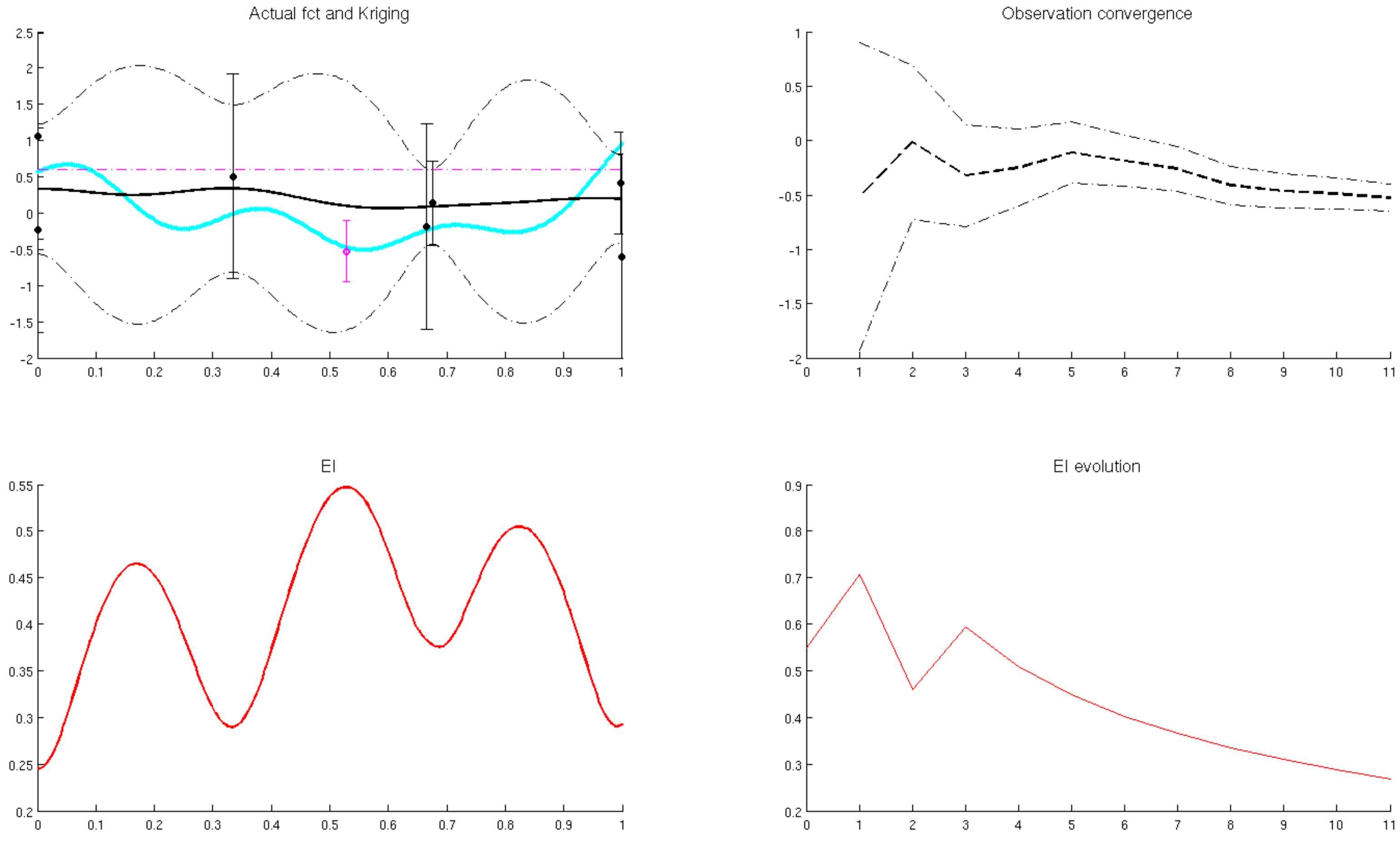
Iteration 2: 1 step used / 91 remaining



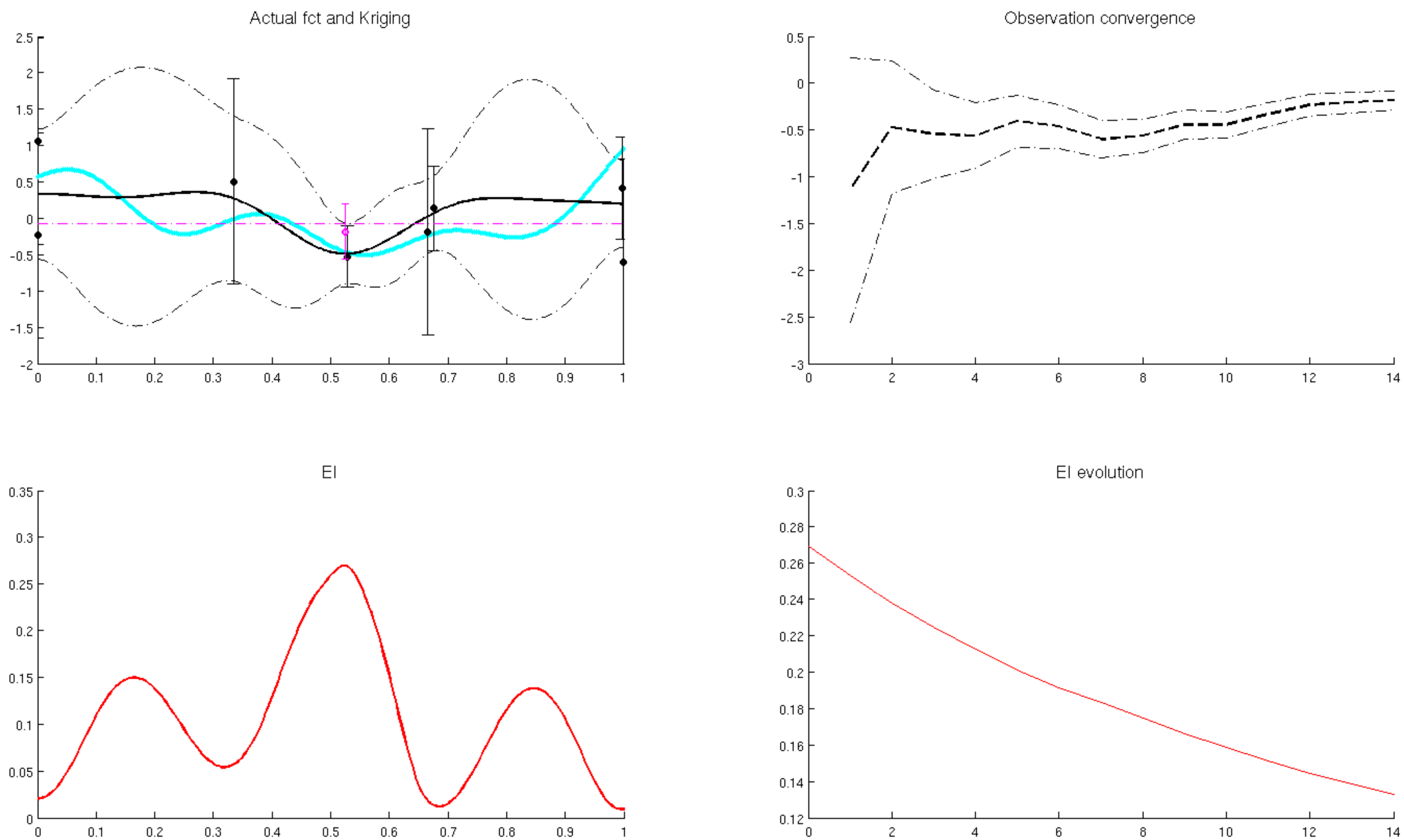
Iteration 3: 6 steps used / 85 remaining



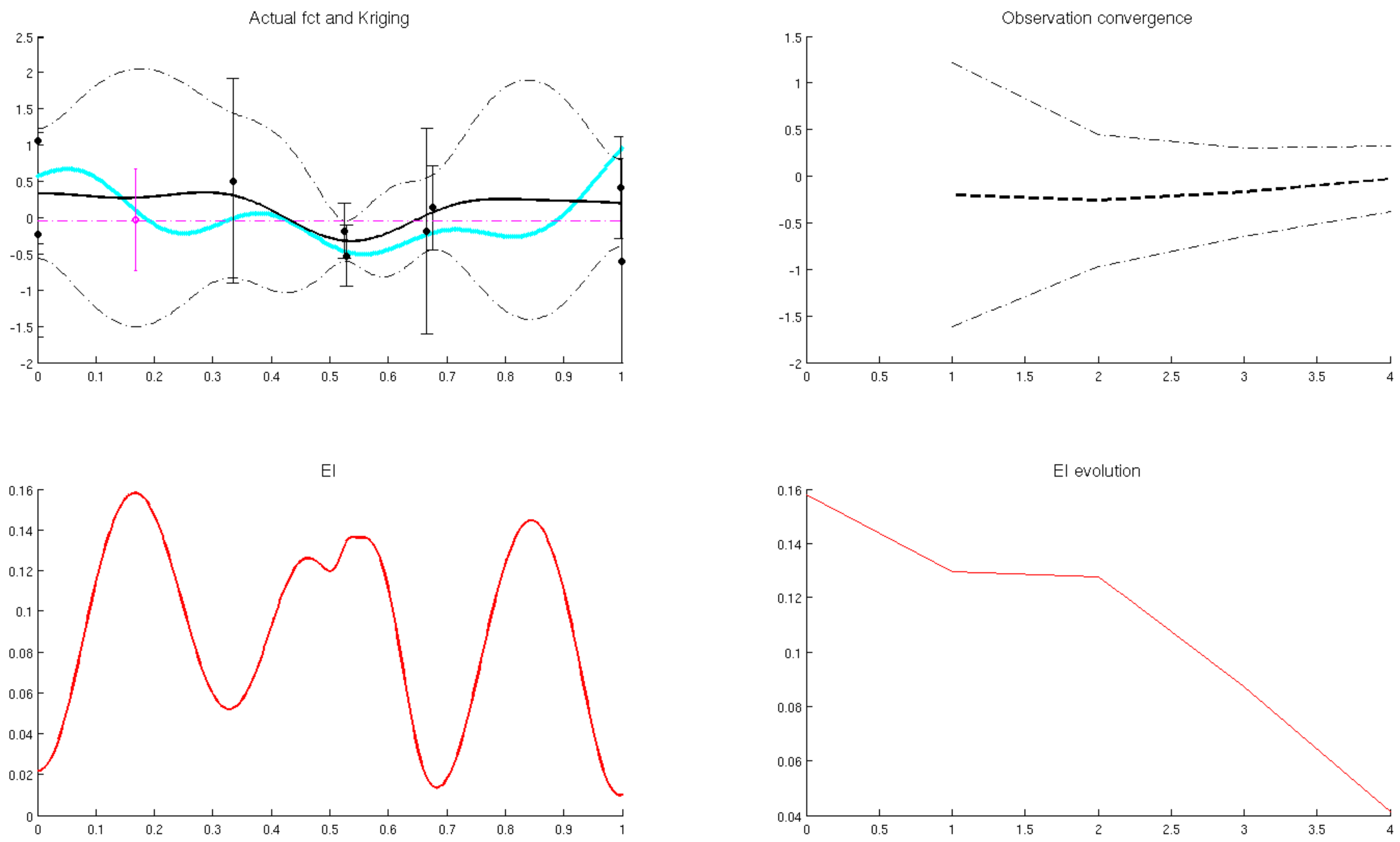
Iteration 4: 11 steps used / 74 remaining



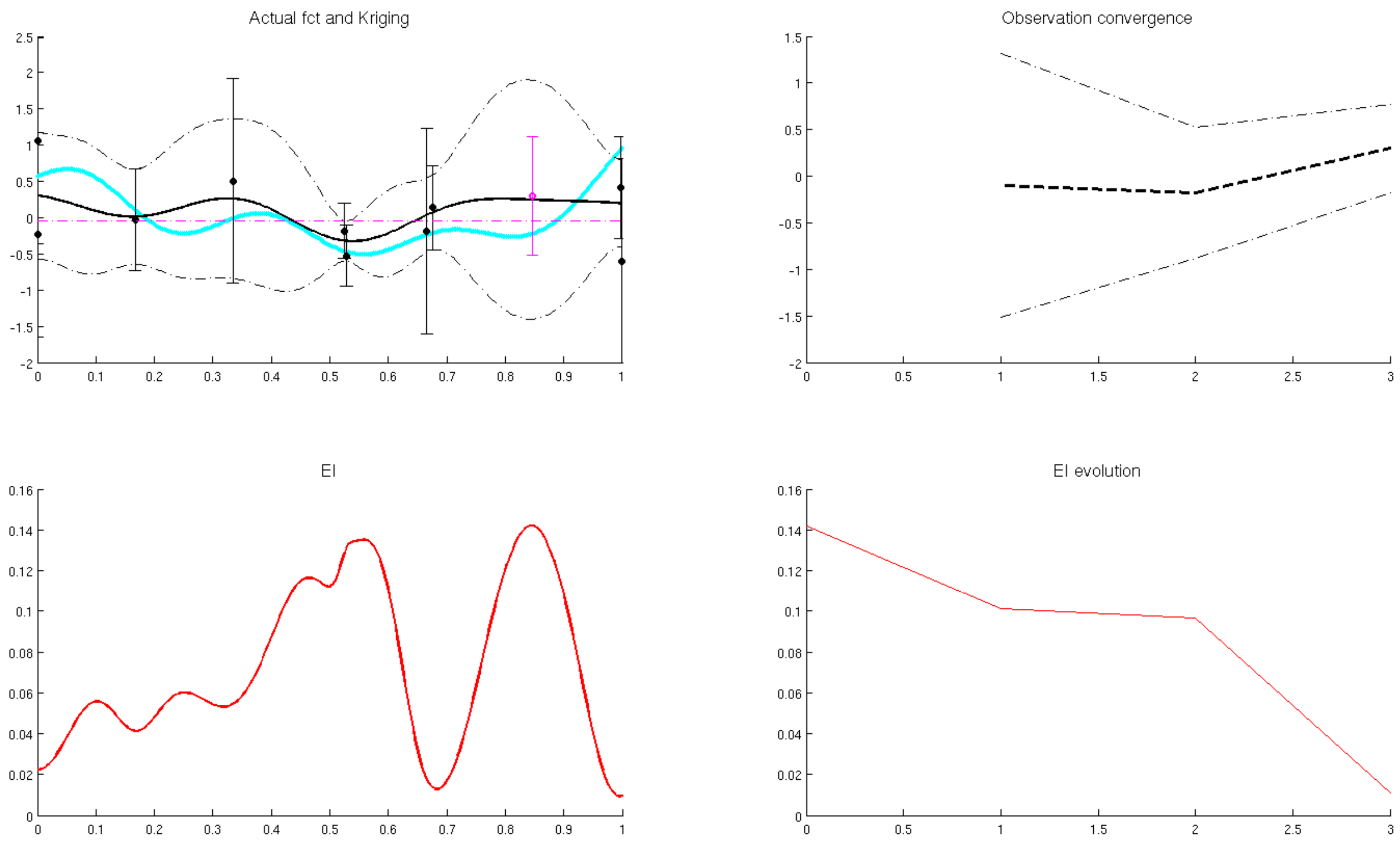
Iteration 5: 14 steps used / 60 remaining



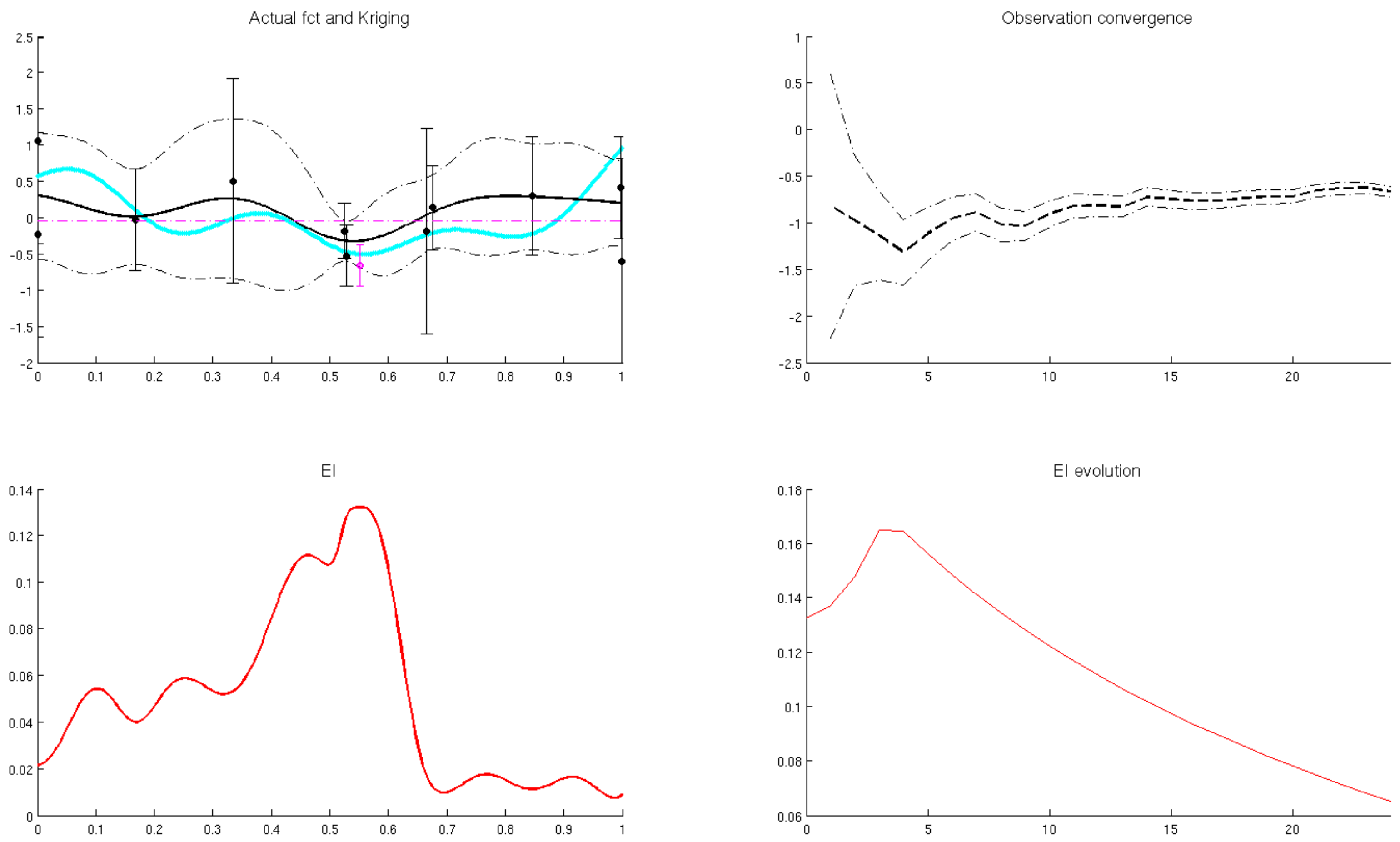
Iteration 6: 4 steps used / 56 remaining



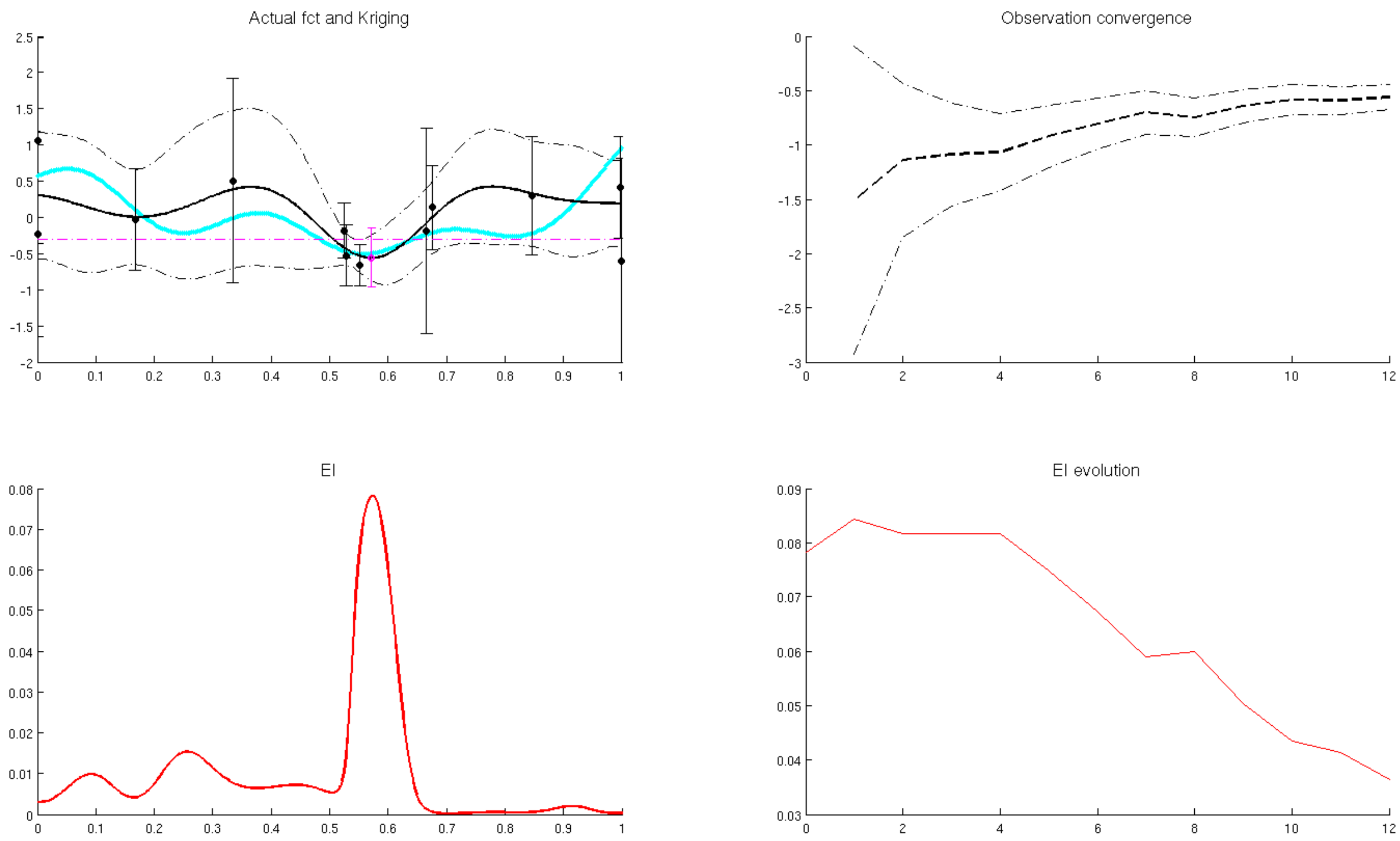
Iteration 7: 3 steps used / 53 remaining



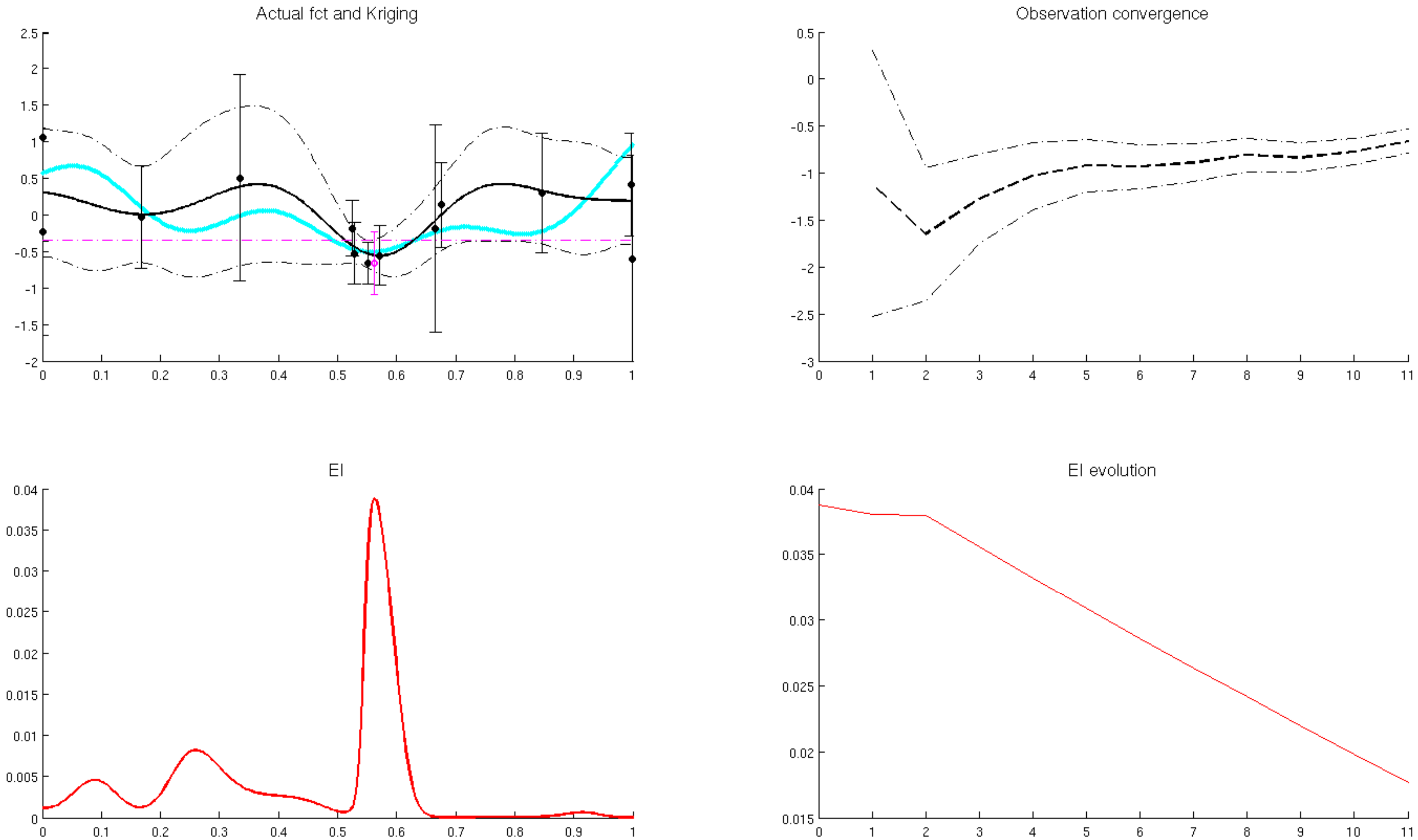
Iteration 8: 22 steps used / 29 remaining



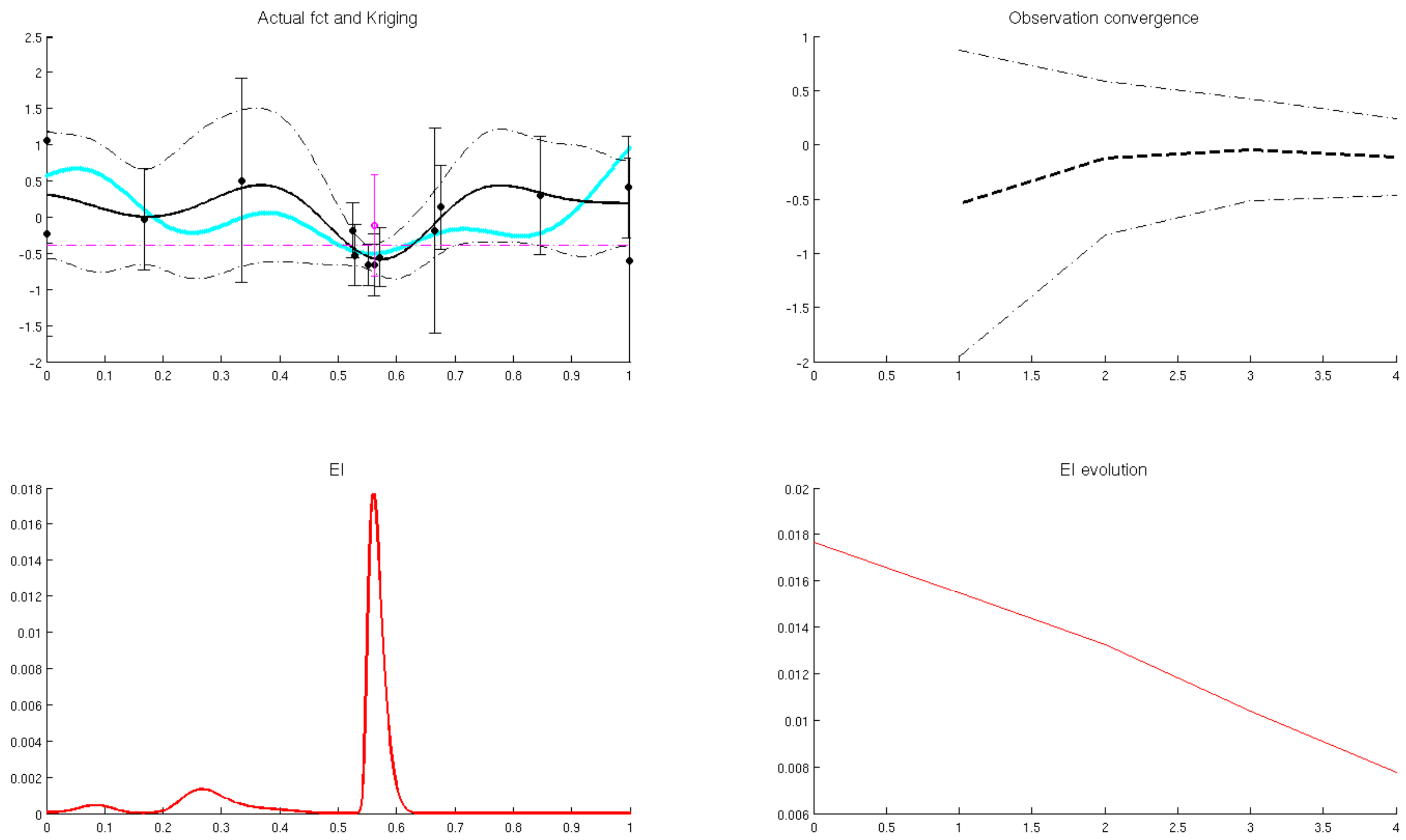
Iteration 9: 12 steps used / 17 remaining



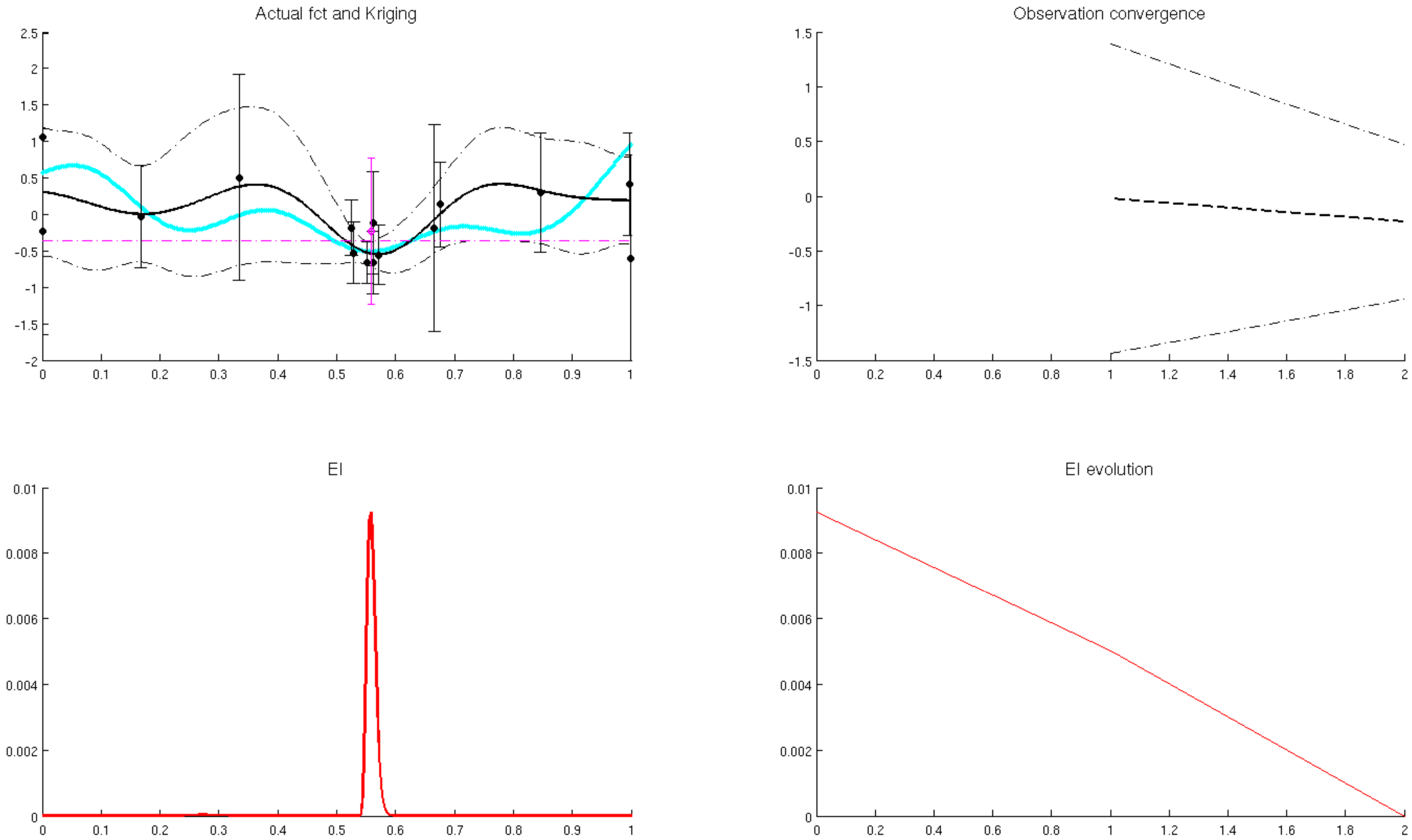
Iteration 10: 11 steps used / 6 remaining



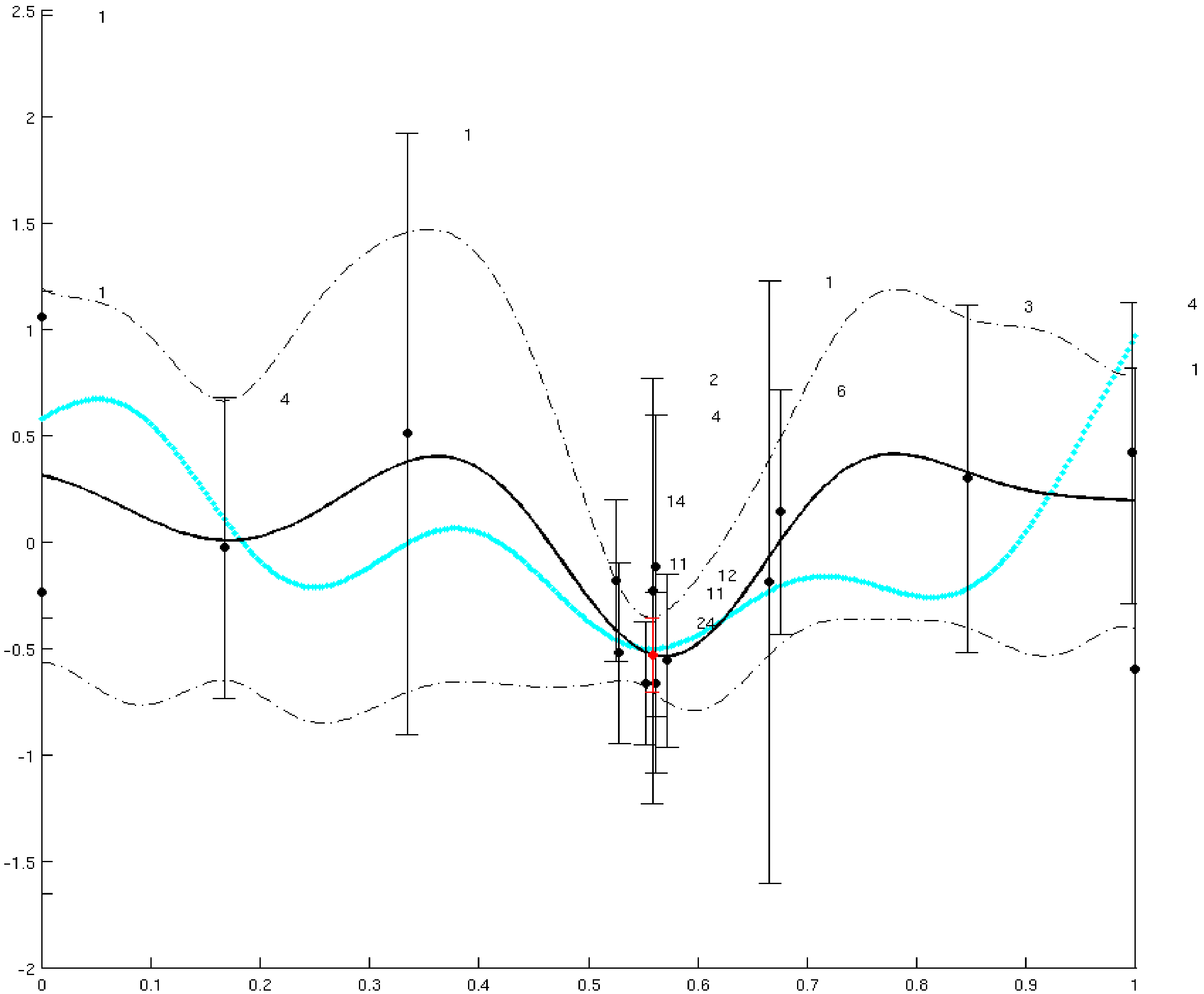
Iteration 11: 4 steps used / 2 remaining



Iteration 12: 2 steps used / 0 remaining



Final DOE and best point



Concluding comments & future work

- Algorithm main features :
  - Decision criterion based on the metamodel
  - Allows on-line resource allocation
  - Takes into account the computational budget

- Limitations
  - Stopping criterion for on-line allocation is empirical
  - Lack of robustness for some configurations

- Next steps :
  - Test on several optimization problems
  - Comparison with other algorithms
  - Adaptability to different error structures

A Failed optimization