Uncertainty Quantification 4

Lecture Notes

Prof. Ian Vernon

Michaelmas Term

1 Introduction (1 lecture)

- Many scientific areas are increasingly employing *Complex Mathematical Models*.
- These models are used to describe complex physical systems.
- For example:
  - Climate science: climate models.
  - Cosmology/astro physics: galaxy formation simulations.
  - Epidemiology: disease models (Covid, HIV, TB, Typhoid, HPV etc.).
  - Nuclear physics: quantum chromodynamic models of nuclear structure.
  - Engineering: Energy Systems (renewables, power networks etc.).
  - Geology: Oil Reservoir simulations.
  - Vulcanology: volcano pyroclastic flow simulations.
  - Systems biology: biochemical reaction networks, root dynamics of crop models.
  - Many more examples...

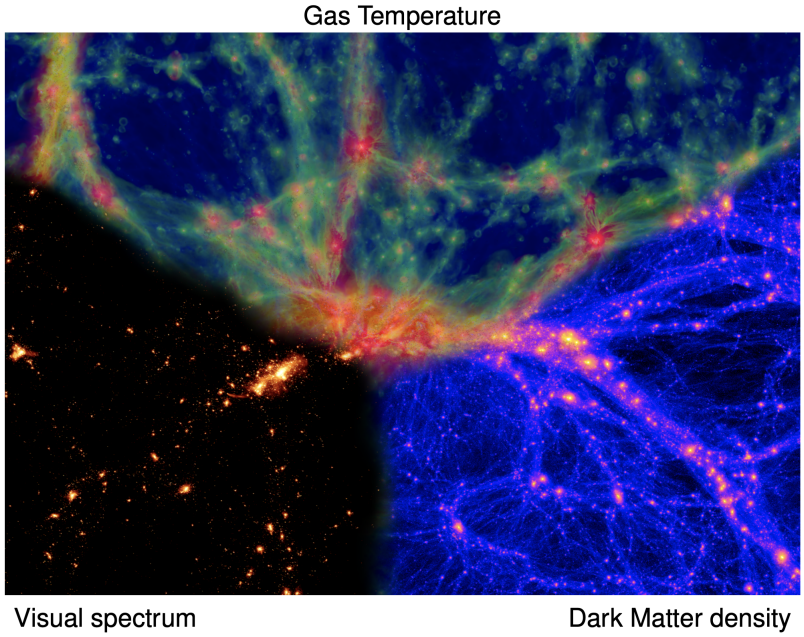


Figure 1: An example of an expensive computer model: the EAGLE model is a simulation of the formation and growth of millions of galaxies from soon after the Big Bang until the current day.

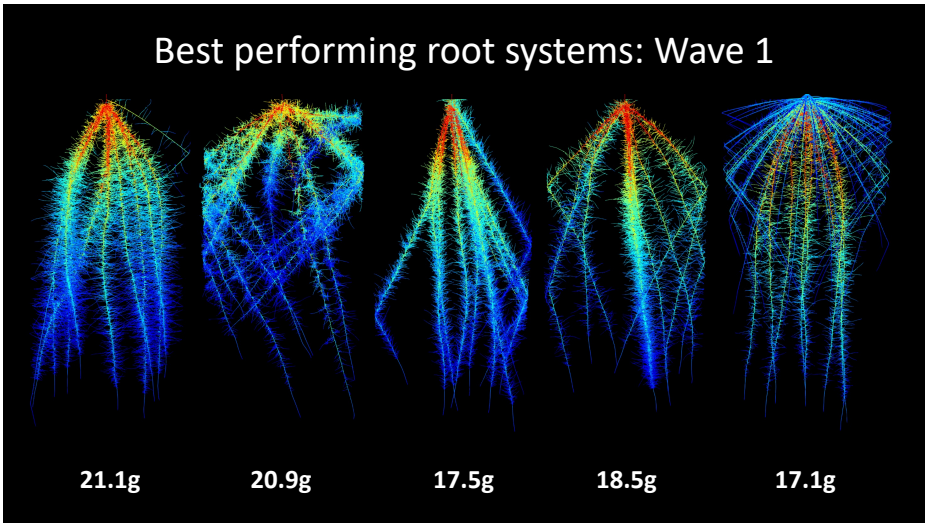


Figure 2: Output from the OpenSimRoot model: an expensive stochastic model simulating the root growth of maize over 42 days. Important to understand to maximise food yield and increase robustness of maize plant to climate change/extreme weather events.

- These *computer models* or *simulators* embody multiple physical laws, processes, relationships and dynamics.
- They are complex: usually in the form of $>$ 10000's lines of code.
- The idea that we can analytically solve them (e.g. using integration tricks, solving PDEs with boundary conditions etc.) is woefully naive.
- They are usually solved using sophisticated combinations of numerical integration techniques which **take a long time** for a single evaluation (and have finite accuracy).
- For example, a galaxy formation simulation typically takes from 1 week to 1 month for a single evaluation. Climate models are similar.
- So what is the problem? Why can't we just wait for a month and then we are done?
- First we have to ask what are these models used for? Many things, but typically 5 major goals:
    1. Scientific understanding (learning what is going on?),
    2. Inference (learning about specific physical quantities),
    3. Forecasting (predicting the future of the system behaviour),
    4. Designing future experiments (for various scientific goals e.g. better inference, better forecasting etc.),
    5. Decision support (endgame: making better decisions about the world).

Major problem 1: The Speed problem

- All of the above desired uses of these models will require multiple evaluations of the simulator (more on this in a second).
- If the simulator takes a long time to run, this will make any "responsible" analysis **utterly infeasible**.
- For example, the simulators typically have many input parameters, the "correct" values of which we do not know.
- We therefore have to explore this input parameter space to find places where the model is "acceptable" e.g. where the model outputs match historical observed data.
- If the model takes a long time to run, this can be incredibly hard!
- E.g. consider exploring a 20-dimensional input space, when it takes 1 week to evaluate a single point...
- Due to uncertainties on observed data, we actually need to find regions of input parameter space, not just single optimal points. Even harder...

Major problem 2: Uncertainties, Uncertainties, Uncertainties

- All of the above desired uses of these models involve multiple sources of uncertainty.
- In order to perform a responsible analysis, we need to identify every major uncertainty and quantify them and their effects: a challenging task.
- E.g. uncertainties on observed data (obvious) but how about the fact that the simulator itself is only an approximation to reality?
- Most scientific groups **fall down badly at this point**.
- Such uncertainties dictate that even more simulator evaluations will be required for a full analysis (in order to explore all corresponding possibilities).

Major Sources of Uncertainty

Identifying and quantifying all uncertainties is vital. These include:

- **Observational Uncertainty**
  - The simulators typically produce large numbers of outputs, each of which we may wish to compare to past data or use to predict the future.
  - May have complex behaviour and a complex and **uncertain** relation to observed data.

  - Observations performed on the real system will not be perfect and lead to many, possibly complex, uncertainties.

- **Structural Model Discrepancy**
  - The **simulator is not real**! As in, it is not a perfect match to the real system (i.e. the real world).
  - Surely, we should take simulator inaccuracy into account in our analysis?
  - Many scientific groups neglect this leading to dangerous errors.

- **Input Parameter Uncertainty**
  - As described above, the simulators typically have many input parameters, representing physical attributes of the system, the "correct" values of which we do not know.
  - We need to incorporate this uncertainty into every subsequent calculation.
  - In fact, is it even meaningful to talk of the "correct" values of the parameters, if the model is not the same as reality? This point raises deep questions about the nature of Bayesian analysis itself.

- **Simulator Evaluation/Emulator Uncertainty**
  - As the simulator takes a long time to run, we will never be able to run it at every point over a (continuous) input space.
  - We will hence always be uncertain (in a subjective sense) about the simulator output except at a small number of evaluation points.
  - We should hence incorporate this evaluation uncertainty into our subsequent calculations.

- **Additional Uncertainties**
  - Many other types of uncertainty depending on context eg.
  - Uncertain initial conditions, forcing functions, boundary conditions, numerical accuracy of the solver/integration scheme, stochasticity of the model, decision uncertainties (utilities, actions) etc.

- All of the above needs to be addressed, otherwise the analysis will be dangerously naive and possibly highly misleading.

- The full uncertainty analysis for complex systems described by slow complex computer models/simulators is therefore needed for responsible use: this (loosely) defines the area of "**Uncertainty Quantification**".

2 Emulation

2.1 Introduction to Emulation

Problem

- We want to learn about an uncertain function (the computer model) which maps (unknown) physical inputs x to outputs of interest $f(x)$, but...
- The above speed problem is *really serious*: why?
- Imagine a computer model/simulator with 20 input dimensions represented as the vector $x \in \mathbb{R}^{20}$ and 1 output dimension represented by $f(x) \in \mathbb{R}$.
- To fully explore the simulator's behaviour (goal 1) we may wish to evaluate it on a 10x10x10... grid.
- But this would require 10^{20} evaluations. If the simulator took just 30 minutes for a single evaluation, this would take 5.71×10^{15} years i.e. 5.71 thousand trillion years: totally infeasible!
- In fact even just exploring the corners of such a space is problematic! There are 2^{20} corners which would take 59.8 years, again assuming a 30 min runtime.
- We will *never* really explore much of the function by direct evaluation!
- In fact, this speed issue utterly prevents us from realizing each of the 5 main goals (as they all require large numbers of evaluations).

Solution

- We build an *emulator*: a fast statistical construct that mimics the computer model function but which is **extremely fast to evaluate**: often several orders of magnitude faster.
- Use the emulator instead of the simulator in the calculations for the 5 main goals.
- The emulator provides a prediction for $f(x)$ and also an *uncertainty statement about this prediction*: this uncertainty can be incorporated into the subsequent analysis, to acknowledge the fact that we have used the emulators (this makes it far better that other methods e.g. interpolators, proxy models etc).
- Emulators are Bayesian constructs and hence exploit our prior knowledge about the simulator e.g. suspected smoothness, differentiability, physical features etc.

Two Types of Emulator

- There are two main types of emulator to mimic $f(x)$ that we shall study:
- Bayes Linear emulators: use second order specifications only. Fast, flexible, second order statements about $f(x)$ i.e. means, variances and covariances. Based on an alternative version of Bayesian Statistics that we shall learn.
- Gaussian Process emulators: full probabilistic specification across functions. Fully Bayesian, so can answer many questions, but does require a full probabilistic specification which we may not be satisfied with, along with more stringent diagnostics.

2.2 Bayes Linear Emulators: Part 1

The Computer Model (or Simulator) of interest $f(x)$

- Let us look at a simple 1D example to get a feel for what is going on.
- Imagine the complex computer model was just the function

$$f(x) = \sin(2\pi x) \tag{1}$$

but **we don't know this!**

- We are interested in the behaviour of $f(x)$ over the range of inputs $x \in [0, 1]$.

Designing Runs of the Simulator $f(x)$

- Imagine we only have the computational resources to evaluate the function at 6 input locations $x^{(j)}, j = 1, \ldots, 6$ (as each evaluation takes a day) and we make the initial choice

$$x_D = (x^{(1)}, \ldots, x^{(6)}) = (0, 0.2, 0.4, 0.6, 0.8, 1) \tag{2}$$

- This means we obtain model output (after 6 days!) at the 6 locations:

$$\begin{aligned} D &= (f(x^{(1)}), f(x^{(2)}), \ldots, f(x^{(6)}))^T \\ &= (\sin(2\pi \times 0), \sin(2\pi \times 0.2), \ldots, \sin(2\pi \times 1))^T \end{aligned} \tag{3}$$

Specifying the Emulator Structure for $f(x)$

- We need to represent our beliefs about the function $f(x)$ otherwise there is not much more we can do.

- We can use a very simple emulator:

$$f(x) \;=\; u(x) \tag{4}$$

where $u(x)$ is a weakly stationary process (you can think of this as an uncertain function) with properties that reflect our prior beliefs about $f(x)$.

- As $u(x)$ is a weakly stationary process we just need to specify its second order structure.

Prior Beliefs about $f(x)$

- So what might we know *a priori* about $f(x)$ (assuming of course that we don't know the actual function)?
- As $f(x)$ has been constructed to mimic a real physical system, and has been built from underlying equations, we may know quite a bit about it e.g.
  - **Location**: We may know roughly where $f(x)$ lies due say to analytic or physical reasons. In our example we may assert the prior statement $\mathrm{E}[f(x)] = 0$.
  - **Variability**: We may know roughly how far away from its mean we expect it to wander. In our example assert $\mathrm{Var}[f(x)] = 0.5^2$ say, so that a cautious 3 sigma interval (for all x) is $\mathrm{E}[f(x)] \pm 3\sqrt{\mathrm{Var}[f(x)]} = [-1.5, 1.5]$.
  - **Wigglyness/smoothness**: We may have some idea of how wiggly or smooth the function will be i.e. if it rapidly changes direction/has many turning points. There are various ways to do this, but a powerful form is via the covariance between unknown f at pairs of points i.e. by specifying $\mathrm{Cov}[f(x), f(x')]$. E.g. in our example assert:

$$\mathrm{Cov}[f(x), f(x')] \;=\; \mathrm{Cov}[u(x), u(x')] \;=\; \sigma^2 \exp\left\{-\frac{|x-x'|^2}{\theta^2}\right\} \tag{5}$$

Here $\sigma = \sqrt{\mathrm{Var}[f(x)]} = 0.5$ as before.
It is worth examining the main features of this covariance specification.

1. Distant points have unrelated outputs, i.e. if x and x' are far away we have:

$$\lim_{|x-x'|\to\infty} \mathrm{Cov}[f(x), f(x')] \;=\; 0 \tag{6}$$

2. Close points have highly correlated outputs, i.e. if x and x' close:

$$\lim_{x\to x'} \mathrm{Cov}[f(x), f(x')] \;=\; \sigma^2 \quad\Longleftrightarrow\quad \lim_{x\to x'} \mathrm{Cor}[f(x), f(x')] \;=\; 1 \tag{7}$$

remembering that $\sigma = \sqrt{\mathrm{Var}[f(x)]} = \sqrt{\mathrm{Var}[f(x')]}$ to get the correlation.

3. The strength of correlation is moderated by the correlation length parameter θ, i.e. for fixed $|x - x'| = a$ we have

$$\lim_{\theta \to 0} \text{Cov}[f(x), f(x')] \;=\; 0 \quad \text{and} \quad \lim_{\theta \to \infty} \text{Cor}[f(x), f(x')] \;=\; 1 \qquad (8)$$

The corrollary of 1., 2. and 3. is that if we perform a run of the model at x' and hence know $f(x')$ this will give us a lot of information about $f(x)$ only if x is close to x' as judged by θ. If x and x' are far apart we will not change our prior beliefs about $f(x)$ in light of $f(x')$.

For our example we specify a reasonable value of $\theta = 0.25$ (a quarter of the range of x).

Updating Our Beliefs about $f(x)$

- We have only given a second order prior specification (means, variances and covariances) and no full joint probability distributions about $f(x)$ so how can we update our beliefs about $f(x)$ in light of the six runs $D = (f(x^{(1)}), f(x^{(2)}), \ldots, f(x^{(6)}))^T$? Bayes Theorem will not help us here...

- There is a whole area of statistics that allows us to do just this, known as "Bayes Linear Statistics" (much more on this later!).

- This provides the following updating formulae for *adjusting our second order beliefs about* $f(x)$ at the new input point x:

$$\text{E}_D[f(x)] \;=\; \text{E}[f(x)] \;+\; \text{Cov}[f(x), D]\text{Var}[D]^{-1}(D - \text{E}[D]) \qquad (9)$$

$$\text{Var}_D[f(x)] \;=\; \text{Var}[f(x)] \;-\; \text{Cov}[f(x), D]\text{Var}[D]^{-1}\text{Cov}[D, f(x)] \qquad (10)$$

where $\text{E}_D[f(x)]$ and $\text{Var}_D[f(x)]$ are our *adjusted expectation and variance for* $f(x)$*, adjusted by data* D. These can be thought of as "similar" to the posterior mean and variance in a standard Bayesian analysis, but they are distinct quantities (more on this later). They are the primary outputs of a Bayes Linear emulator.

- These quantities $\text{E}_D[f(x)]$ and $\text{Var}_D[f(x)]$ **are incredibly fast to calculate** requiring only a single matrix inverse and some fast linear algebra. This is why emulators are often several orders of magnitude faster that the simulator (e.g. for Galaxy formation models we achieved a $10^9 - 10^{12}$ speed up!).

- All quantities on the right hand side of equations (9) and (10) can now be calculated from the above prior second order specifications.

- Note: where did these BL update formula come from? We will derive them soon, but lets use them first to see what they can do.

- We have from above that $\mathrm{E}[f(x)] = 0$ and $\mathrm{Var}[f(x)] = \sigma^2 = 0.5^2$.
- For the vector of six runs D we have that a priori (before we have evaluated them):

$$\mathrm{E}[D] \;=\; \mathrm{E}[(f(x^{(1)}), \ldots, f(x^{(6)}))^T] \;=\; (0,0,0,0,0,0)^T \tag{11}$$

- However $\mathrm{Var}[D]$ is now a 6×6 covariance matrix with individual entries given using equation (5):

$$\begin{aligned} \mathrm{Var}[D]_{jk} &= \mathrm{Cov}[f(x^{(j)}), f(x^{(k)})] && (12)\\ &= \sigma^2 \exp\left\{-\frac{|x^{(j)} - x^{(k)}|^2}{\theta^2}\right\} && (13) \end{aligned}$$

  (with $x^{(j)} = (0, 0.2, 0.4, 0.6, 0.8, 1)$, $\theta = 0.25$ and $\sigma = 0.5$ as specified above).
- Finally we need the covariance between the function/simulator $f(x)$ at the new input point x and the 6 runs in D. This is a length 6 row vector with elements again given by equation (5):

$$\begin{aligned} \mathrm{Cov}[f(x), D]_j &= \mathrm{Cov}[f(x), f(x^{(j)})] && (14)\\ &= \sigma^2 \exp\left\{-\frac{|x - x^{(j)}|^2}{\theta^2}\right\} && (15) \end{aligned}$$

- E.g. if we want to emulate at the new input point $x = 0.25$ we can find the adjusted expectation of $f(0.25)$ adjusted by data D, using equation (9):

$$\begin{aligned} \mathrm{E}_D[f(0.25)] &= \mathrm{E}[f(0.25)] \;+\; \mathrm{Cov}[f(0.25), D]\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D])\\ &= 0 \;+\; \left(0.5^2 \exp\left\{-\frac{|0.25-0|^2}{0.25^2}\right\}, \ldots, 0.5^2 \exp\left\{-\frac{|0.25-1|^2}{0.25^2}\right\}\right) \times\\ &\quad \begin{pmatrix} 0.5^2 \exp\left\{-\frac{|0-0|^2}{0.25^2}\right\} & \ldots & 0.5^2 \exp\left\{-\frac{|0-1|^2}{0.25^2}\right\} \\ \vdots & \ddots & \vdots \\ 0.5^2 \exp\left\{-\frac{|1-0|^2}{0.25^2}\right\} & \ldots & 0.5^2 \exp\left\{-\frac{|1-1|^2}{0.25^2}\right\} \end{pmatrix}^{-1} \left(\begin{pmatrix} f(x^{(1)}) \\ \vdots \\ f(x^{(6)}) \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \right)\\ &= \left(\exp\{-1\}, \ldots, \exp\{-3^2\}\right) \begin{pmatrix} 1 & \ldots & \exp\{-4^2\} \\ \vdots & \ddots & \vdots \\ \exp\{-4^2\} & \ldots & 1 \end{pmatrix}^{-1} \begin{pmatrix} \sin(2\pi \times 0) \\ \vdots \\ \sin(2\pi \times 1) \end{pmatrix}\\ &= 1.0223 \qquad \text{(to 5 s.f.)} && (16) \end{aligned}$$

  where, except for the case of very low run number we would perform this calculation on a computer e.g. using R/RStudio.

- Similarly we have the adjusted variance of $f(0.25)$ adjusted by data D, using equation (10):

$$\begin{aligned}
\mathrm{Var}_D[f(0.25)] &= \mathrm{Var}[f(0.25)] \;-\; \mathrm{Cov}[f(0.25), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, f(0.25)] \\
&= 0.5^2 \;-\; \left(0.5^2 \exp\left\{-\frac{|0.25-0|^2}{0.25^2}\right\}, \ldots, 0.5^2 \exp\left\{-\frac{|0.25-1|^2}{0.25^2}\right\}\right) \times \\
&\quad \begin{pmatrix} 0.5^2 \exp\left\{-\frac{|0-0|^2}{0.25^2}\right\} & \ldots & 0.5^2 \exp\left\{-\frac{|0-1|^2}{0.25^2}\right\} \\ \vdots & \ddots & \vdots \\ 0.5^2 \exp\left\{-\frac{|1-0|^2}{0.25^2}\right\} & \ldots & 0.5^2 \exp\left\{-\frac{|1-1|^2}{0.25^2}\right\} \end{pmatrix}^{-1} \begin{pmatrix} 0.5^2 \exp\left\{-\frac{|0.25-0|^2}{0.25^2}\right\} \\ \vdots \\ 0.5^2 \exp\left\{-\frac{|0.25-1|^2}{0.25^2}\right\} \end{pmatrix} \\
&= 0.5^2 \;-\; \left(\exp\{-1\}, \ldots, \exp\{-3^2\}\right) \begin{pmatrix} 1 & \ldots & \exp\{-4^2\} \\ \vdots & \ddots & \vdots \\ \exp\{-4^2\} & \ldots & 1 \end{pmatrix}^{-1} \begin{pmatrix} \exp\{-1\} \\ \vdots \\ \exp\{-3^2\} \end{pmatrix} \\
&= 0.0026192 \qquad \text{(to 5 s.f.)}
\end{aligned}$$

- A 3-sigma prediction interval can be made for $f(x = 0.25)$ of

$$\begin{aligned}
\mathrm{E}_D[f(0.25)] \;\pm\; 3\sqrt{\mathrm{Var}_D[f(0.25)]} &= 1.0223 \pm 3\sqrt{0.0026192} \\
&= 1.0223 \pm 0.15353 \\
&= [0.8688, 1.176] \qquad \text{(to 4 s.f.)} \qquad (17)
\end{aligned}$$

  which includes the true value of $f(x = 0.25) = \sin(2\pi \times 0.25) = \sin(\pi/2) = 1$.

- However, predicting $f(x)$ at *a single point* is a little boring! How about instead we predict $f(x)$ at 201 x points covering the range $[0, 1]$ given by the vector:

$$x_P \;=\; (0, 0.005, 0.01, 0.015, \cdots, 0.99, 0.995, 1) \qquad (18)$$

  and then plot the 201 adjusted expectations $\mathrm{E}_D[f(x_P)]$ against x_P and the 201 prediction intervals $\mathrm{E}_D[f(x_P)] \;\pm\; 3\sqrt{\mathrm{Var}_D[f(x_P)]}$. Figure 3 shows this (in each case the 201 points have been joined together with lines).

- (Note that there are shortcuts in how to calculate several points at once in the BL update equations: we will see this when we derive the BL equations in section 2.3).

- There are several things to note about Figure 3:

  1. The adjusted expectation $\mathrm{E}_D[f(x)]$ (blue line) does a decent job of mimicking the true $f(x) = \sin(2\pi x)$ function (black line), but it is of course not perfect, as it is based on limited information (6 runs and our prior views).

2. The adjusted expectation $\mathrm{E}_D[f(x)]$ perfectly interpolates the six known runs (green points) i.e. it goes through them.
3. The adjusted variance $\mathrm{Var}_D[f(x)]$ is sophisticated in that it is x dependent: it is larger in places that are far from known runs, and smaller close to runs.
4. In fact we see that the prediction interval $\mathrm{E}_D[f(x_P)] \pm 3\sqrt{\mathrm{Var}_D[f(x_P)]}$ contains the true function (it doesn't always have to!), but also shrinks to zero size at the known run locations.
5. This is because we are mimicking a deterministic (i.e. repeatable) simulator, and hence once we have seen a run at $x^{(1)}$ say, we are now sure of the value of $f(x^{(1)})$ from now on and this is reflected in the adjusted variance $\mathrm{Var}_D[f(x)]$ dropping to zero at these points.
6. This is a very welcome and important ability of emulators. Note the contrast to say a regression model which assumes each measurement has error attached: the regression prediction rarely interpolates points and never has zero error anywhere.
7. This feature links (as most statistics does eventually!) to our interpretation of uncertainty. We take the subjective Bayesian viewpoint: there is nothing "random" about the function $f(x)$, we simply do not know its value at most values of x except for the six values we have evaluated in D. The uncertainty is therefore purely due to our lack of knowledge and hence the subjective viewpoint is very natural and powerful here.

Adding further runs of $f(x)$

- Say we are interested in when low values of $f(x)$ occur, and have the computational resources to perform two more runs.
- We can look at figure 3 and see that if we add two more runs at $x = 0.7$ and $x = 0.9$ this may improve our emulator for low values.
- So we now extend x_D and D to be

$$\begin{aligned} x_D &= (x^{(1)}, \ldots, x^{(8)}) = \{0, 0.2, 0.4, 0.6, 0.8, 1, 0.7, 0.9\} && (19) \\ D &= (f(x^{(1)}), f(x^{(2)}), \ldots, f(x^{(8)}))^T && (20) \\ &= (\sin(2\pi \times 0), \sin(2\pi \times 0.2), \ldots, \sin(2\pi \times 0.9))^T \end{aligned}$$

- Note the order of points in x_D does not matter (as long as they are consistent with the order in D of course!).
- Figure 4 shows this new emulator trained on 8 input points.

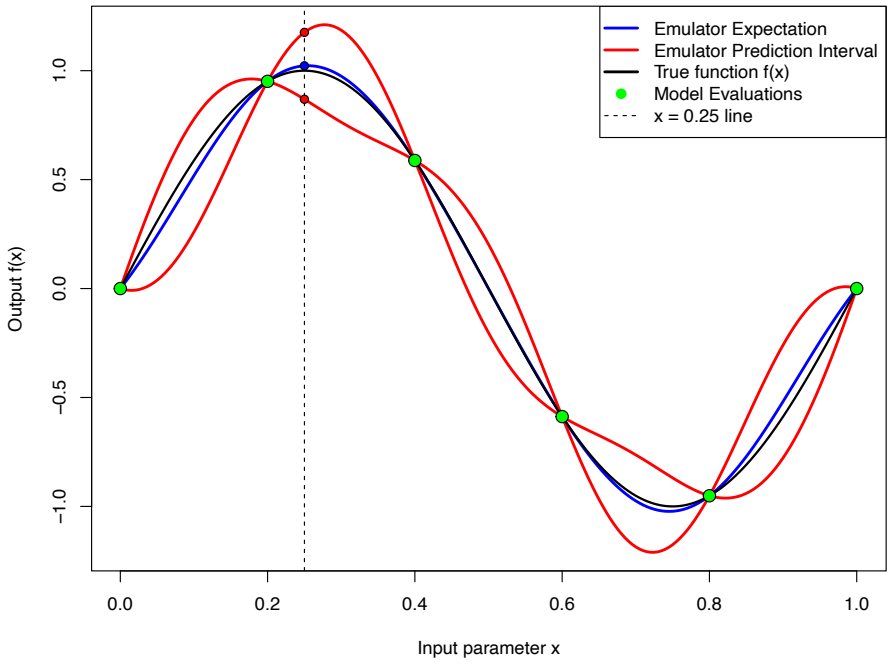


Figure 3: Simple 1D emulator of the $f(x) = \sin(2\pi x)$ function using only 6 runs. The blue line gives the emulator expectation $\mathrm{E}_D[f(x)]$ and red lines give the emulator prediction interval $\mathrm{E}_D[f(x)] \pm 3\sqrt{\mathrm{Var}_D[f(x)]}$. The vertical black dashed line is at $x = 0.25$, where we made the first emulator evaluation given in equations (16) and (17). Here $\theta = 0.25$ and $\sigma = 0.5$.

- We see that it is far more accurate for low values of $f(x)$ and the prediction intervals tightly match the true $f(x) = \sin(2\pi x)$ function.
- This approach of including batches of additional runs (chosen using the current emulator) is called **Sequential Design** in UQ/statistics (and sometimes called **Active Learning** in Machine Learning). It is incredibly powerful, as it prevents us wasting time doing many runs in uninteresting parts of the input space, which for expensive computer models is vital.

Investigating BL Emulator Parameters and Properties

- The emulator equations (9) and (10) along with the covariance specification (5) can seem a little mysterious at first.
- To improve our intuition, we will now investigate some of their properties, specifically

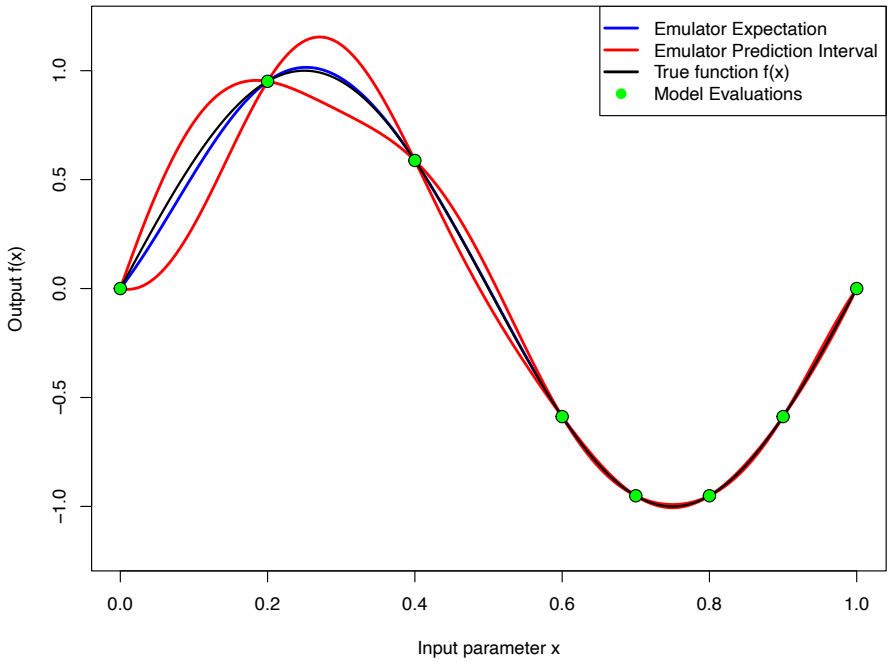


Figure 4: Simple 1D emulator of the $f(x) = \sin(2\pi x)$ function now using 8 runs, with two new runs added to target lower values of $f(x)$. The blue line gives the emulator expectation $\mathrm{E}_D[f(x)]$ and the red lines give the emulator prediction interval $\mathrm{E}_D[f(x)] \pm \sqrt{\mathrm{Var}_D[f(x)]}$. Here $\theta = 0.25$ and $\sigma = 0.5$.

what happens when we change parts of our prior specification e.g.

1. The prior variance $\sigma^2 = \mathrm{Var}[f(x)]$.
2. The correlation length θ.
3. The prior expectation $\mathrm{E}[f(x)]$.

Varying the prior variance $\sigma^2 = \mathrm{Var}[f(x)]$

- Figure 5 shows different emulator prediction intervals for different values of prior variance $\sigma^2 = \mathrm{Var}[f(x)]$.
- We see immediately that the adjusted expectation $\mathrm{E}_D[f(x)]$ is unchanged by altering σ. This is perhaps unexpected, but is due to the symmetries of the emulator specification (see problem sheet for a proof).

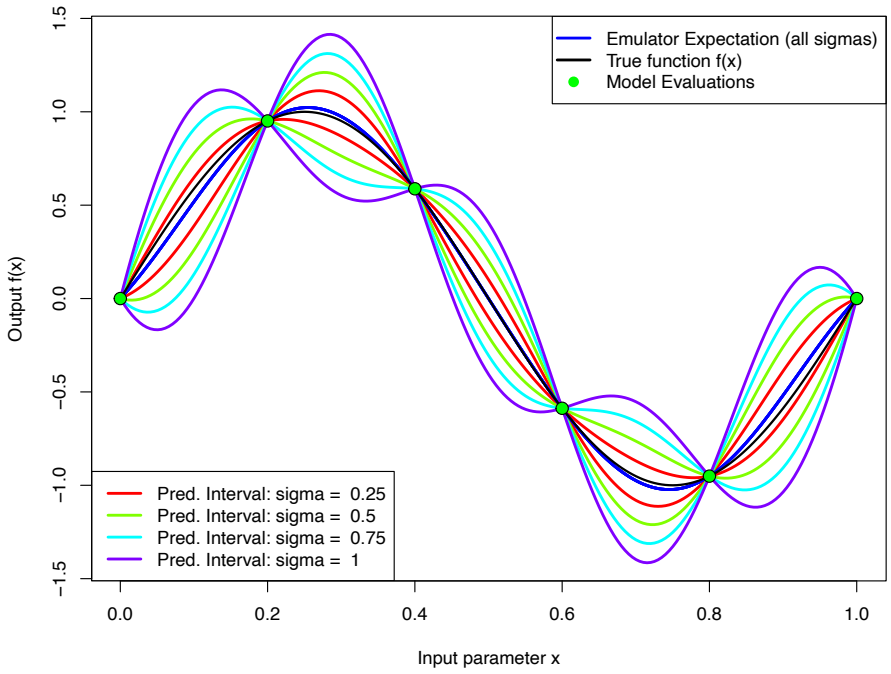
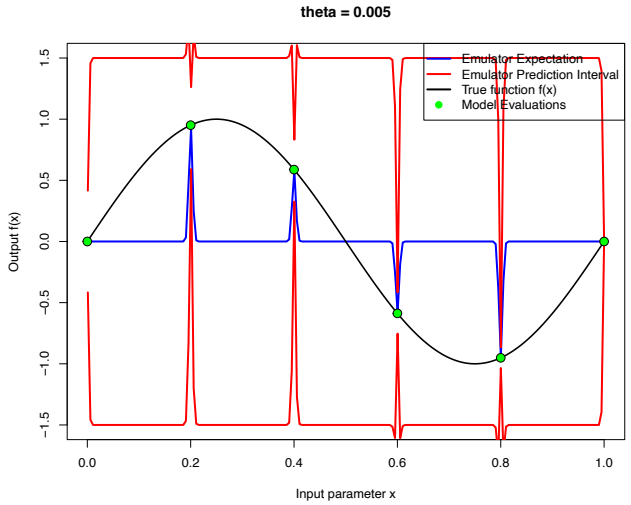


Figure 5: Simple 1D emulator of the $f(x) = \sin(2\pi x)$ function using 6 runs, but now varying the prior variance σ^2. Different emulator prediction intervals $\mathrm{E}_D[f(x)] \pm \sqrt{\mathrm{Var}_D[f(x)]}$ are shown. Note that the emulator expectation $\mathrm{E}_D[f(x)]$ is the same regardless of the value of σ. Here $\theta = 0.25$.

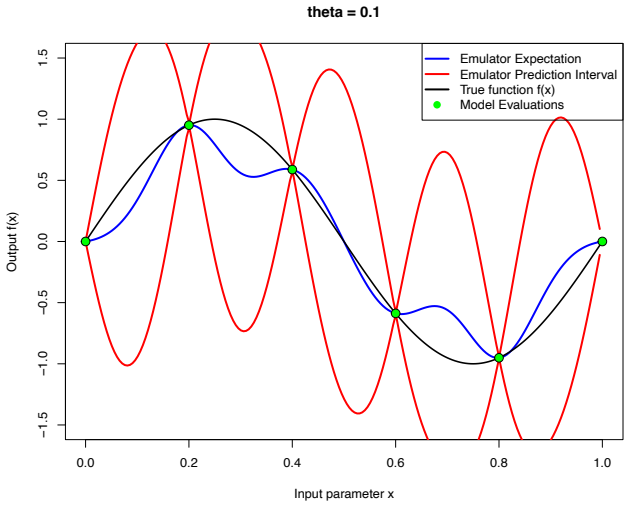
- Second, we see that the adjusted variance $\mathrm{Var}_D[f(x)]$ seems to increase linearly with the prior variance. This is a little more expected: if we are more unsure to begin with, then we are more unsure afterwards. (See problem sheet for a proof of this).

Varying the correlation length θ

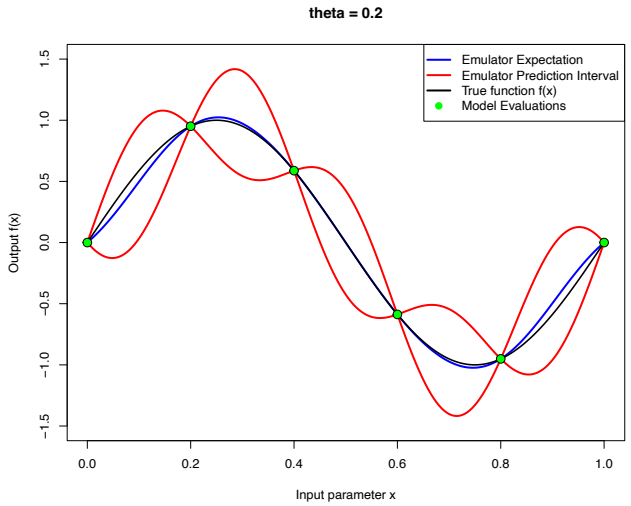
- Figure 6 shows different emulator prediction intervals for 6 different values of the correlation length θ.
- Refer to the covariance specification equation (5): we see for small values of θ known runs only have an influence on an area close to them, while far from known runs the emulator resorts to the prior expectation and variance so the prediction interval tends to $\mathrm{E}[f(x)] \pm 3\sqrt{\mathrm{Var}[f(x)]}$.
- For larger values of θ, known runs have a much larger influence and lead to the emulator being very sure (i.e. low adjusted variance), and possibly too sure in some



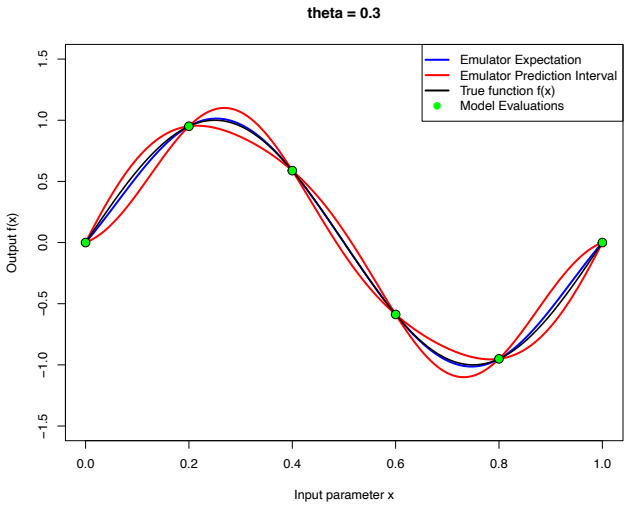
(a) Simple emulator with $\theta = 0.005$, $\sigma = 0.5$.



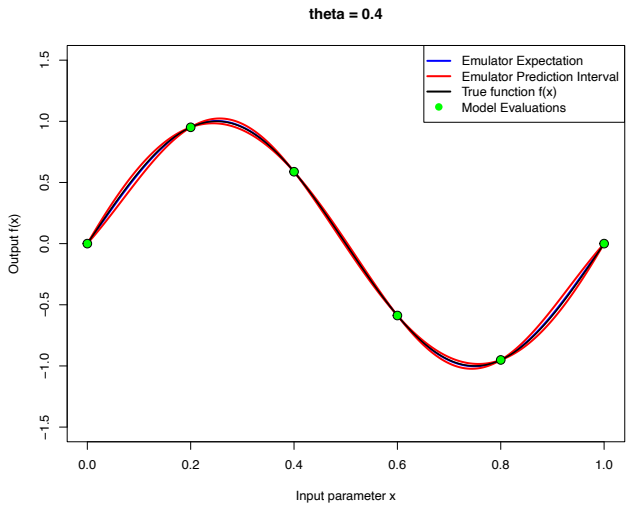
(b) Simple emulator with $\theta = 0.1$, $\sigma = 0.5$.



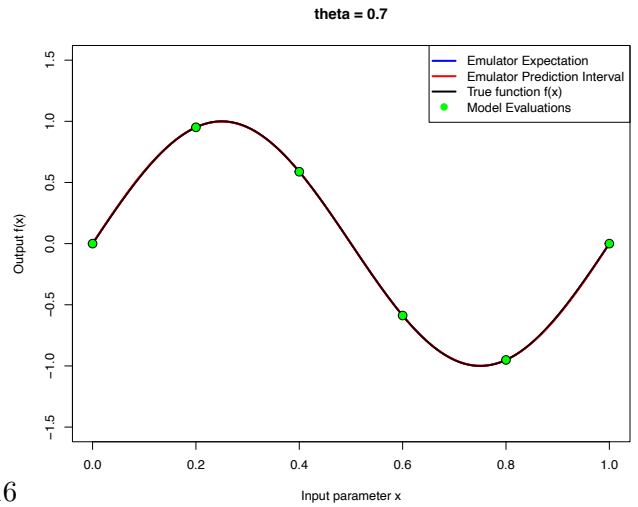
(c) Simple emulator with $\theta = 0.2$, $\sigma = 0.5$.



(d) Simple emulator with $\theta = 0.3$, $\sigma = 0.5$.



(e) Simple emulator with $\theta = 0.4$, $\sigma = 0.5$.



(f) Simple emulator with $\theta = 0.7$, $\sigma = 0.5$.

Figure 6: The effect of varying the correlation length parameter θ on emulator output.

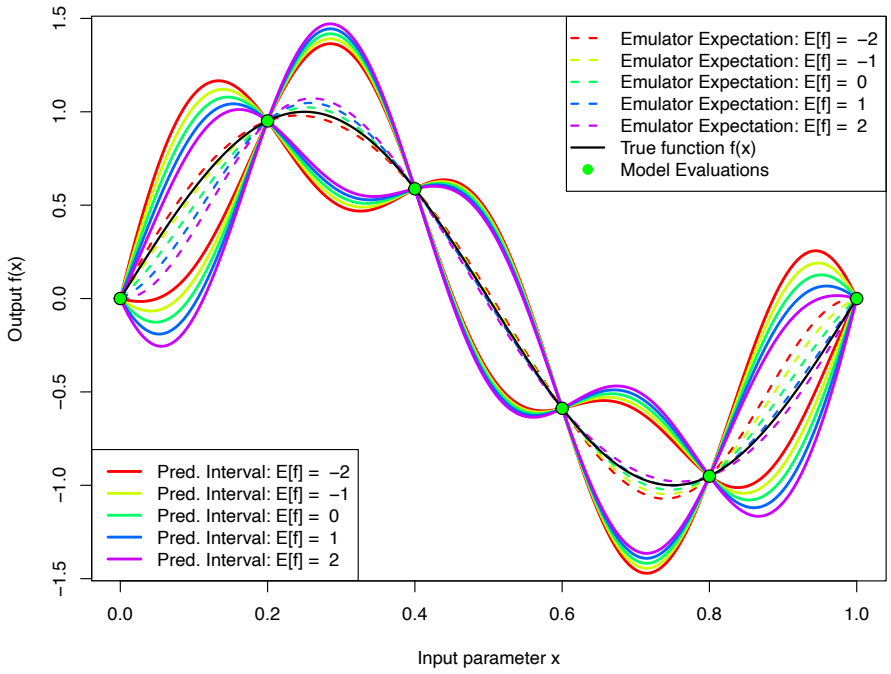


Figure 7: Simple 1D emulator of the $f(x) = \sin(2\pi x)$ function using 6 runs, but now varying the prior variance σ^2. Different emulator prediction intervals $\mathrm{E}_D[f(x)] \pm \sqrt{\mathrm{Var}_D[f(x)]}$ are shown. Note that the emulator expectation $\mathrm{E}_D[f(x)]$ is the same regardless of the value of σ. Here $\theta = 0.25$.

cases, although here it is doing very well at mimicking the $f(x) = \sin(2\pi x)$ function.

Varying the prior expectation $\mathrm{E}[f(x)]$

- Figure 7 shows different emulator prediction intervals for different values of the prior expectation $\mathrm{E}[f(x)]$.
- We don't see so much change, except perhaps toward the edges/boundaries.
- However, if we were to reduce θ and then change $\mathrm{E}[f(x)]$ we would see a more noticeable effect, or if we were to evaluate the emulator outside of the range of the runs $[0, 1]$ (see computer class).

Summary so far

- Emulators mimic slow computer models (which could take hours/days/weeks etc.) but they are **extremely fast** to evaluate. They provide predictions of $f(x)$ in the form of $\mathrm{E}_D[f(x)]$ along with uncertainty statements $\mathrm{Var}_D[f(x)]$ in a fraction of a second.
- We have developed a simple 1-dimensional emulator which is fine, but 2 dimensions and general d dimensions are far more fun. We will do this soon, but first we need to justify the Bayes Linear update equations (9) and (10) with a brief summary of Bayes Linear methods.

2.3 Bayes Linear Methods

Difficulties with the Full Bayesian Approach

- The fully Bayesian paradigm is powerful, general and meaningful, and is now widely deployed. E.g. it has more meaning and hence worth that the Frequentist approach.
- However, working within the Bayesian framework, we can encounter certain difficulties when considering multivariate analyses.
    1. Even in small problems, it can be difficult to distil all of our prior knowledge into a satisfactory and meaningful joint prior probability specification;
    2. Given such a specification, the computations for learning from data, particularly in areas such as experimental design, become technically difficult and extremely computer intensive (e.g. using MCMC);
    3. Further, in higher-dimensions the likelihood surface tends to be very complicated, so that full Bayes calculations may become highly non-robust (i.e small perturbations in our specification or data could lead to quite different posterior beliefs).
- Therefore if, in complex problems, we are unable to make and analyse full prior probability specifications, it follows that we require methods based around simpler belief specifications.

de Finetti: Expectation as Primitive

- de Finetti spent most of his life studying subjective conceptions of probability, and founded the subjective Bayesian paradigm.
- However, he actually proposed the use of Expectation as the primitive entity to base any uncertainty analysis on, as opposed to Probability.
- In the Bayes Linear approach we follow de Finetti and take Expectation as *primitive*.
- Probabilities (where relevant) enter as derived quantities: they are the Expectations of Indicator functions e.g. we have the indicator function for event A:

$$\mathbb{1}_A \;=\; \begin{cases} 1 & \text{if A true} \\ 0 & \text{if A false} \end{cases} \tag{21}$$

  and if we know the expectation of $\mathbb{1}_A$ then we can derive the probability of A as

$$P[A] \;=\; \mathrm{E}[\mathbb{1}_A] \tag{22}$$

  which demonstrates that here, probability would be a derived quantity and no longer a fundamental (primitive) one.

- Note this asymmetry: if probability is treated as the primitive quantity then one has to specify (in the continuous case) an infinite set of probabilities in order to derive a single expectation.

Belief Specification

- The Bayes Linear approach is of course subjectivist, and therefore in any analysis we need to specify our beliefs over all random quantities of interest.
- However, as we are treating Expectation as primitive, the necessary objects that we are required to specify are just the Expectations, Variances and Covariances of the random quantities of interest.
- (If we have beliefs about higher orders e.g. skew, kurtosis etc. we can include these in the analysis too!)
- For example, say we are interested in predicting the vector of two random quantities $B = (B_1, B_2)^T$ from knowledge of two other random quantities $D = (D_1, D_2)^T$ which we will measure soon, then all we need to specify are $\mathrm{E}[B]$, $\mathrm{E}[D]$, $\mathrm{Var}[B]$, $\mathrm{Var}[D]$ and $\mathrm{Cov}[B, D]$.

Adjusted Expectation and Variance

- We are interested in how our beliefs about $\mathrm{E}[B_1]$ and $\mathrm{E}[B_2]$ change in the light of information given by measuring/observing D_1 and D_2.
- We look among the collection of linear estimates, i.e. those of form $c_0 + c_1 D_1 + c_2 D_2$, and choose constants c_0, c_1, c_2 to minimise the prior expected squared error loss in estimating each of B_1 and B_2, e.g. for B_1 we minimise:

$$\mathrm{E}\big[[B_1 - c_0 - c_1 D_1 - c_2 D_2]^2\big]. \tag{23}$$

Theorem: The Bayes Linear Adjusted Expectation $\mathrm{E}_D[B]$

The linear estimates for B_1 and B_2 that minimise the prior expected squared error loss are given by $\mathrm{E}_D[B] = (\mathrm{E}_D[B_1], \mathrm{E}_D[B_2])^T$ where

$$\mathrm{E}_D[B] = \mathrm{E}[B] + \mathrm{Cov}[B, D]\mathrm{Var}[D]^{\dagger}(D - \mathrm{E}[D]). \tag{24}$$

which we refer to as the *adjusted expectation* for collection B given collection D.

- The "†" symbol denotes the generalised inverse (e.g. the Moore-Penrose generalized inverse) which allows for degeneracy i.e. some zero eigenvalues in the $\mathrm{Var}[D]$ matrix.

- Note that we can therefore update a vector of random quantities B simultaneously: very useful for emulation when we update $f(x)$ at many input points x_P at once.

Proof: The Bayes Linear Adjusted Expectation $\mathrm{E}_D[B]$

We assume that we have specified, a priori, all second order quantities for B and D i.e. we have $\mathrm{E}[B]$, $\mathrm{E}[D]$, $\mathrm{Var}[B]$, $\mathrm{Var}[D]$ and $\mathrm{Cov}[B, D]$. Set T as the above prior expected squared error loss when using linear estimate $c_0 + c_1 D_1 + c_2 D_2$ to estimate B_1:

$$T = \mathrm{E}\left[[B_1 - c_0 - c_1 D_1 - c_2 D_2]^2\right] \tag{25}$$

and choose c_0, c_1, c_2 to minimise T. First we differentiate w.r.t. c_0 and set the result equal to zero:

$$\frac{\partial T}{\partial c_0} = \mathrm{E}\left[2[B_1 - c_0 - c_1 D_1 - c_2 D_2](-1)\right] = 0 \tag{26}$$

$$\Leftrightarrow \quad c_0 = \mathrm{E}[B_1] - c_1 \mathrm{E}[D_1] - c_2 \mathrm{E}[D_2] \tag{27}$$

We can now replace this expression for c_0 (equation (27)) back into T (equation (25)) (note slight subtleties here but fine as c_0 linear in c_1 and c_2) to get a more intuitive expression:

$$\begin{aligned} T &= \mathrm{E}\left[[B_1 - c_0 - c_1 D_1 - c_2 D_2]^2\right] && (28)\\ &= \mathrm{E}\left[[B_1 - (\mathrm{E}[B_1] - c_1 \mathrm{E}[D_1] - c_2 \mathrm{E}[D_2]) - c_1 D_1 - c_2 D_2]^2\right] && (29)\\ &= \mathrm{E}\left[[(B_1 - \mathrm{E}[B_1]) - c_1(D_1 - \mathrm{E}[D_1]) - c_2(D_2 - \mathrm{E}[D_2])]^2\right] && (30) \end{aligned}$$

Differentiating by c_1 and setting to zero gives:

$$\begin{aligned} \frac{\partial T}{\partial c_1} = \mathrm{E}\Big[-2\big\{(B_1 - \mathrm{E}[B_1]) - c_1(D_1 - \mathrm{E}[D_1]) - c_2(D_2 - \mathrm{E}[D_2])\big\}(D_1 - \mathrm{E}[D_1])\Big] &= 0\\ \Leftrightarrow \quad \mathrm{E}\big[(B_1 - \mathrm{E}[B_1])(D_1 - \mathrm{E}[D_1])\big] - c_1 \mathrm{E}\big[(D_1 - \mathrm{E}[D_1])^2\big] &\\ -c_2 \mathrm{E}\big[(D_2 - \mathrm{E}[D_2])(D_1 - \mathrm{E}[D_1])\big] &= 0\\ \Leftrightarrow \quad \mathrm{Cov}[B_1, D_1] - c_1 \mathrm{Var}[D_1] - c_2 \mathrm{Cov}[D_2, D_1] &= 0 \end{aligned} \tag{31}$$

a constraint in terms of the prior second order quantities which we have. A similar calculation for c_2 yields:

$$\begin{aligned} \frac{\partial T}{\partial c_2} &= 0\\ \Leftrightarrow \quad \mathrm{Cov}[B_1, D_2] - c_1 \mathrm{Cov}[D_1, D_2] - c_2 \mathrm{Var}[D_2] &= 0 \end{aligned} \tag{32}$$

With an eye on a more general proof (for any length of D) we can combine constraints (31) and (32) into a single (row) vector constraint:

$$\begin{aligned}
\mathrm{Cov}\left[B_1, \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}\right] - \begin{pmatrix} c_1 & c_2 \end{pmatrix} \begin{pmatrix} \mathrm{Var}[D_1] & \mathrm{Cov}[D_1, D_2] \\ \mathrm{Cov}[D_2, D_1] & \mathrm{Var}[D_2] \end{pmatrix} &= 0 && (33)\\
\Leftrightarrow \quad \mathrm{Cov}[B_1, D] - \begin{pmatrix} c_1 & c_2 \end{pmatrix} \mathrm{Var}[D] &= 0 && (34)\\
\Leftrightarrow \quad \begin{pmatrix} c_1 & c_2 \end{pmatrix} &= \mathrm{Cov}[B_1, D]\mathrm{Var}[D]^\dagger && (35)
\end{aligned}$$

With this we can construct the form of the adjusted expectation for B_1

$$\begin{aligned}
\mathrm{E}_D[B_1] &\equiv c_0 + c_1 D_1 + c_2 D_2 && (36)\\
&= \mathrm{E}[B_1] + c_1(D_1 - \mathrm{E}[D_1]) + c_2(D_2 - \mathrm{E}[D_2]) && (37)\\
&= \mathrm{E}[B_1] + \begin{pmatrix} c_1 & c_2 \end{pmatrix} \begin{pmatrix} D_1 - \mathrm{E}[D_1] \\ D_2 - \mathrm{E}[D_2] \end{pmatrix} && (38)\\
&= \mathrm{E}[B_1] + \begin{pmatrix} c_1 & c_2 \end{pmatrix} (D - \mathrm{E}[D]) && (39)\\
\Rightarrow \quad \mathrm{E}_D[B_1] &= \mathrm{E}[B_1] + \mathrm{Cov}[B_1, D]\mathrm{Var}[D]^\dagger (D - \mathrm{E}[D]) && (40)
\end{aligned}$$

Noticing that we would get a similar result for $\mathrm{E}_D[B_2]$ we can represent both results, using the vector form $\mathrm{E}_D[B] = (\mathrm{E}_D[B_1], \mathrm{E}_D[B_2])^T$ as the core Bayes linear update:

$$\mathrm{E}_D[B] = \mathrm{E}[B] + \mathrm{Cov}[B, D]\mathrm{Var}[D]^\dagger (D - \mathrm{E}[D]) \tag{41}$$

which completes the proof.

- The *adjusted version* of B given D is the 'residual' vector

$$\mathbb{A}_D[B] = B - \mathrm{E}_D[B].$$

- We partition the vector B as the sum of two uncorrelated vectors:

$$B = \mathrm{E}_D[B] + \mathbb{A}_D[B],$$

  with $\mathrm{Cov}[\mathrm{E}_D[B], \mathbb{A}_D[B]] = 0$ (see problem sheet for proof).

- Hence we can partition the variance matrix of B into 2 variance components

$$\begin{aligned}
\mathrm{Var}[B] &= \mathrm{Var}[\mathrm{E}_D[B]] + \mathrm{Var}[\mathbb{A}_D[B]] && (42)\\
&= \mathrm{RVar}_D[B] + \mathrm{Var}_D[B] && (43)
\end{aligned}$$

  These are the *resolved variance matrix* and the *adjusted variance matrix*.

- The resolved variance $\mathrm{RVar}_D[B] = \mathrm{Var}[\mathrm{E}_D[B]]$ will be resolved once we observe D (note that the only random quantity in $\mathrm{E}_D[B]$ is D), however even after observing D the adjusted variance $\mathrm{Var}_D[B] = \mathrm{Var}[\mathbb{A}_D[B]]$ will remain.
- Hence, when we actually observe D we 'adjust' our prior variance $\mathrm{Var}[B]$ to become the adjusted variance $\mathrm{Var}_D[B] = \mathrm{Var}[\mathbb{A}_D[B]]$.

Theorem: Adjusted and Resolved Variance Matrices $\mathrm{Var}_D[B]$ and $\mathrm{RVar}_D[B]$

The two variance matrices are calculated as (see problem sheet for proof):

$$\begin{aligned} \mathrm{Var}_D[B] &= \mathrm{Var}[B] - \mathrm{Cov}[B, D]\mathrm{Var}[D]^\dagger\mathrm{Cov}[D, B], \\ \mathrm{RVar}_D[B] &= \mathrm{Cov}[B, D]\mathrm{Var}[D]^\dagger\mathrm{Cov}[D, B]. \end{aligned}$$

- Our variance matrices must be non-negative definite with $0 < \mathbf{tr}\{\cdot\} < \infty$. So, they might be singular but must reflect at least one linear combination with positive variance. That is, $0 \leq a^T\Sigma a < \infty \; \forall a$, and $a^T\Sigma a > 0$ for some a.

Resolution and Adjusted Covariance

- We summarize the value of data D for any quantity B by a scale-free measure which we call the *resolution* of B induced by D,

$$\mathrm{R}_D[B] = 1 - \frac{\mathrm{Var}_D[B]}{\mathrm{Var}[B]} = \frac{\mathrm{Var}[\mathrm{E}_D[B]]}{\mathrm{Var}[B]}.$$

- The resolution lies between 0 and 1, and in general, small (large) resolutions imply that the information has little (much) linear predictive value, given the prior specification.
- The *adjusted covariance matrix* and *resolved covariance matrix* are defined similarly:

$$\mathrm{Cov}_D[B_1, B_2] = \mathrm{Cov}[B_1, B_2] - \mathrm{Cov}[B_1, D]\mathrm{Var}[D]^\dagger\mathrm{Cov}[D, B_2], \quad (44)$$

$$\mathrm{RCov}_D[B_1, B_2] = \mathrm{Cov}[B_1, D]\mathrm{Var}[D]^\dagger\mathrm{Cov}[D, B_2]. \quad (45)$$

- Note that the full adjusted variance formula actually already incorporates the above adjusted covariance (in its off diagonal terms).
- Note also that full Bayesian updating where the random variables have Normal priors leads to the same formulae.

- Note also the links to classical statistical methodology. If B, D are Multivariate Normal, then

$$\mathrm{Var}[B|D] \;=\; \mathrm{Var}[B] - \mathrm{Cov}[B,D]\mathrm{Var}[D]^{\dagger}\mathrm{Cov}[D,B]$$

  is the conditional variance of B given D.

Bayes Linear Diagnostics

- One data has been observed (first for D, then for B) the Bayes linear framework has a rich variety of diagnostic tools designed to pick up unexpected observations of D or B but also unexpected changes in our beliefs $\mathrm{E}_D[B]$ upon observing D.

Prior Diagnostics for Observation of $D = d$.

- We can check to see if our observation of $D = d$ is consistent with our prior using a choice of diagnostics.
- The univariate diagnostics are

$$S(d_i) \;=\; \frac{d_i - \mathrm{E}[D_i]}{\sqrt{\mathrm{Var}[D_i]}}$$

  while the multivariate is

$$\mathrm{Dis}(d) \;=\; (d - \mathrm{E}[D])^T \mathrm{Var}[D]^{\dagger}(d - \mathrm{E}[D])$$

- Note $\mathrm{E}[S(D_i)] = 0$ and $\mathrm{Var}[S(D_i)] = 1$, so if we have $S(d_i)$ greater than about 3 this suggests an inconsistency.
- We have that $\mathrm{E}[\mathrm{Dis}(D)] = \mathbf{rank}\{\mathrm{Var}[D]\}$, however $\mathrm{Var}[\mathrm{Dis}(d)]$ is more complex and requires further specification. Large $\mathrm{Dis}(d)$ will of course also suggest inconsistencies.
- These diagnostics will be useful to inform us regarding problems with our prior emulator specification.

Diagnostics for the Observed Adjustment $\mathrm{E}_d[B]$

- We may replace $D - \mathrm{E}[D]$ in our formula for the adjusted expectation by the observed value $d - \mathrm{E}[D]$ to obtain:

$$\mathrm{E}_d[B] \;=\; \mathrm{E}[B] + \mathrm{Cov}[B,D]\mathrm{Var}[D]^{\dagger}(d - \mathrm{E}[D]).$$

  Which we refer to as the *observed adjusted expectation*. Note that in computer model emulation we often don't make a clear notational distinction between D and d).

- We should check how different the observed adjusted expectation is from its prior expectation, standardized with respect to the variance of the adjusted expectation.
- We calculate the univariate *Standardized Adjustments*:

$$S(\mathrm{E}_d[B_i]) \;=\; \frac{\mathrm{E}_d[B_i] - \mathrm{E}[\mathrm{E}_D[B_i]]}{\sqrt{\mathrm{Var}[\mathrm{E}_D[B_i]]}} \;=\; \frac{\mathrm{E}_d[B_i] - \mathrm{E}[B_i]}{\sqrt{\mathrm{RVar}_D[B_i]}}$$

  and the multivariate *Adjustment Discrepancy*:

$$\mathrm{Dis}_d(B) \;=\; (\mathrm{E}_d[B] - \mathrm{E}[B])^T \mathrm{RVar}_D[B]^\dagger (\mathrm{E}_d[B] - \mathrm{E}[B])$$

- Again $\mathrm{E}[S(\mathrm{E}_d[B_i])] = 0$, $\mathrm{Var}[S(\mathrm{E}_d[B_i])] = 1$ and $\mathrm{E}[\mathrm{Dis}_d(B)] = \mathbf{rank}\{\mathrm{RVar}_D[B]\}$ so large values warrant further investigation.
- These diagnostics are less familiar but show if our beliefs have been moved in suspicious ways by the observation of d e.g. even if the prior diagnostics regarding d are borderline, we may find that $S(\mathrm{E}_d[B_i])$ and $\mathrm{Dis}_d(B)$ are still reasonable implying that our beliefs have only been altered in more modest ways, and we may still be willing to proceed.

Final Observation Diagnostics for Observation of $B = b$.

- At some point we may actually observe values $B = b$ and in addition to checking how these deviate from the prior expected values $\mathrm{E}[B]$, we should also check the change from adjusted expectation $\mathrm{E}_d[B]$ to actual observation b.
- The appropriate univariate standardized change is

$$S_d(b_i) \;=\; S(\mathbb{A}_d[b_i]) \;=\; \frac{b_i - \mathrm{E}_d[B_i]}{\sqrt{\mathrm{Var}_D[B_i]}}$$

  while the multivariate version is

$$\mathrm{Dis}_d(b) \;=\; (b - \mathrm{E}_d[B])^T \mathrm{Var}_D[B]^\dagger (b - \mathrm{E}_d[B])$$

- Again $\mathrm{E}[S_d(B_i)] = 0$ and $\mathrm{Var}[S_d(B_i)] = 1$ and $\mathrm{E}[\mathrm{Dis}_d(B)] = \mathbf{rank}\{\mathrm{Var}_D[B]\}$, so large values warrant further investigation.
- These final observation diagnostics will be very important for checking the predictive behaviour of emulators (in fact these diagnostics have several names which we will mention!).
- Note how in each case we should really check univariate and multivariate diagnostics: the latter checks for acceptable behaviour across collections of random quantities.

2.4 Bayes Linear Emulators: Part II

- So we have created a 1D emulator and then justified the core Bayes linear update formula as a suitable Bayesian update for a second order specification.
- We will now construct a 2D emulator (as emulating in more dimensions is more fun!).
- Consider the problem of mimicking the following scalar function $f(x)$ of a 2D input $x = (x_1, x_2)$:

$$f(x) \quad = \quad f(x_1, x_2) \quad = \quad -\sin(2\pi x_2) + 0.9\sin(2\pi(1-x_1)(1-x_2)) \tag{46}$$

  and again suppose that

  - evaluating $f(x)$ was extremely expensive, and we only had resources to perform say 16 evaluations.
  - We are interested in the behaviour of $f(x)$ over the region

$$x_1 \in [0,1] \quad \text{and} \quad x_2 \in [0,1] \tag{47}$$

  - We know $f(x)$ is smooth and deterministic (i.e. repeatable).

Generalising an Emulator to 2 Dimensions

- How do we generalise our 1D emulator from before into a 2D emulator?
- Actually the generalisation of a simple emulator to higher dimensions is very simple.
- The only things that change are the design and the covariance matrix construction (very slightly)! The core BL update equations remain the same as the BL update does not "care" which dimension we are in, it only cares about the covariance structure.
- Our design is now a list of 16 points, each of which are 2-dimensional. Lets choose a grid design to start with of $\{0.05, 0.35, 0.65, 0.95\} \times \{0.05, 0.35, 0.65, 0.95\}$ (we shall discuss why this is often a bad choice later on).
- We now arrange x_D as a 16×2 matrix with each row as $x^{(i)}$ a 2D vector, the coordinates of a run, with $i = 1, \ldots, 16$. For our grid design we have:

$$x_D = \begin{pmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(15)} \\ x^{(16)} \end{pmatrix} = \begin{pmatrix} 0.05 & 0.05 \\ 0.35 & 0.05 \\ 0.65 & 0.05 \\ 0.95 & 0.05 \\ 0.05 & 0.35 \\ 0.35 & 0.35 \\ 0.65 & 0.35 \\ 0.95 & 0.35 \\ 0.05 & 0.65 \\ 0.35 & 0.65 \\ 0.65 & 0.65 \\ 0.95 & 0.65 \\ 0.05 & 0.95 \\ 0.35 & 0.95 \\ 0.65 & 0.95 \\ 0.95 & 0.95 \end{pmatrix} \quad (48)$$

- We evaluate the simulator at each of the 16 locations and get output:

$$\begin{aligned} D &= (f(x^{(1)}), \ldots, f(x^{(16)}))^T && (49) \\ &= \big(-\sin(2\pi \times 0.05) + 0.9\sin\big[2\pi(1-0.05)(1-0.05)\big], \ldots, && (50) \\ &\quad -\sin(2\pi \times 0.95) + 0.9\sin\big[2\pi(1-0.95)(1-0.95)\big]\big)^T \end{aligned}$$

- Now we specify similar prior statements to the 1D case by saying again that $f(x)$ is a weakly stationary process:

$$f(x) = u(x) \quad (51)$$

and that our covariance structure is

$$\mathrm{Cov}[f(x), f(x')] = \mathrm{Cov}[u(x), u(x')] = \sigma^2 \exp\left\{-\frac{||x-x'||^2}{\theta^2}\right\} \quad (52)$$

with the only generalisation being that we now use the full Euclidean distance $||x-x'||$ between pairs of 2D vector input points.

- We choose $\mathrm{E}[f(x)] = 0$, $\sigma = \sqrt{\mathrm{Var}[f(x)]} = 1$ and $\theta = 0.45$ as prior emulator quantities.

- We can now construct the length 16 column vector $\mathrm{E}[D]$, the 16×16 matrix $\mathrm{Var}[D]$ and the length 16 row vector $\mathrm{Cov}[f(x), D]$ as usual.

- We can then use the core BL update emulator equations (9) and (10) to find the all important adjusted expectation and variance: $\mathrm{E}_D[f(x)]$ and $\mathrm{Var}_D[f(x)]$.

Visualising a 2D Emulator

- Visualising the emulator output in 2D becomes more tricky/fun: figure 8a shows the adjusted expectation $\mathrm{E}_D[f(x)]$ over $[0,1]^2$ given the 16 runs in a grid design, and figure 8c gives the adjusted variance $\mathrm{Var}_D[f(x)]$, where the set of emulator evaluation points x_P is now a 50×50 grid spanning $[0,1]^2$.
- Figure 8b shows the true function $f(x)$ for comparison.
- The adjusted expectation again agrees precisely with the true function at the 16 run locations, and does a seemingly good job of mimicking $f(x)$ over the input space.
- Note that the adjusted variance shows the local impact of each of the 16 design points: around each we are very sure of what is going on.

Emulator Diagnostics

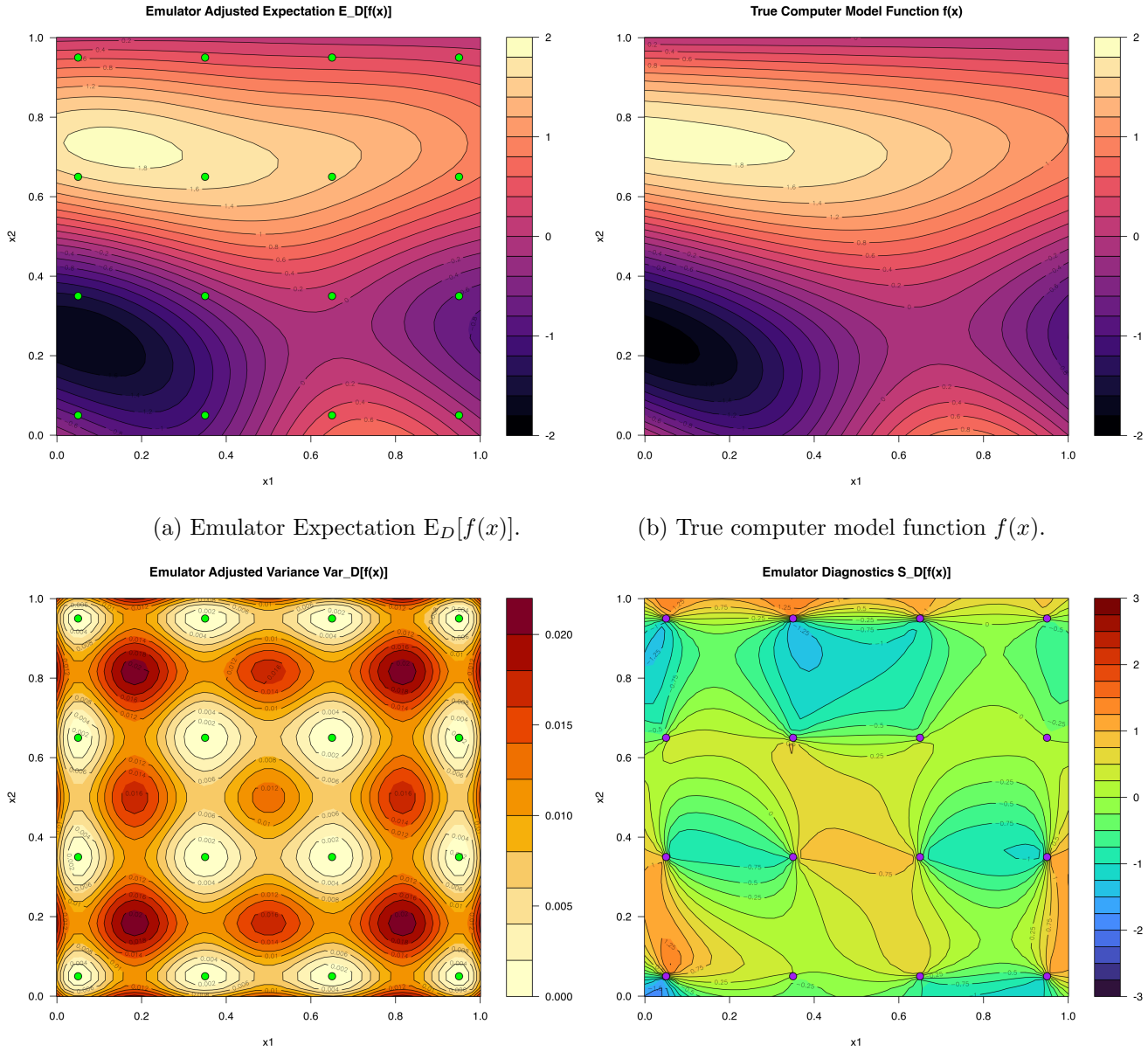
- But how can we tell if the emulator is actually performing well? In 1D we saw if the true $f(x)$ lay within the prediction interval. This leads us onto emulator diagnostics (a topic we may discuss more later) but for now, as we have access to the true function $f(x)$, we can look at the third BL diagnostic:

$$S_D(f(x)) \quad = \quad \frac{f(x) - \mathrm{E}_D[f(x)]}{\sqrt{\mathrm{Var}_D[f(x)]}} \tag{53}$$

  (where we drop the distinction between observed $d = D$).
- Take a moment to look at this: it just compares the true function $f(x)$ with our prediction $\mathrm{E}_D[f(x)]$, scaled by our uncertainty $\sqrt{\mathrm{Var}_D[f(x)]}$.
- If the true $f(x)$ is within the 3-sigma prediction interval then equivalently we will have $-3 < S_D(f(x)) < 3$. As we know $f(x)$ everywhere for this toy example (usually we don't!) we can plot $S_D(f(x))$ over the input space, as given in figure 8d. We can see that the emulator is performing well over all of the input space.

Varying the Emulator Prior Parameters

- We can vary the emulator prior parameters $\mathrm{Var}[f(x)] = \sigma^2$ and $\mathrm{E}[f(x)]$ as before with predictable effects.
- Results of varying the correlation length θ are given in figure 9 and show the increased/decreased effects of each run on their vicinity due to longer/shorter correlation lengths.
- Only when these effects overlap to cover the input space do we get a reasonable emulator expectation $\mathrm{E}_D[f(x)]$ that mimics $f(x)$ well.



(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.

(b) True computer model function $f(x)$.

(c) Emulator Variance $\mathrm{Var}_D[f(x)]$.

(d) Emulator diagnostics $S_D[f(x)]$.

Figure 8: A two dimensional emulator using a 16 point grid design. The locations of the design points i.e. the 16 runs are given as the green points in (a) and (c) and as the purple points in (d).

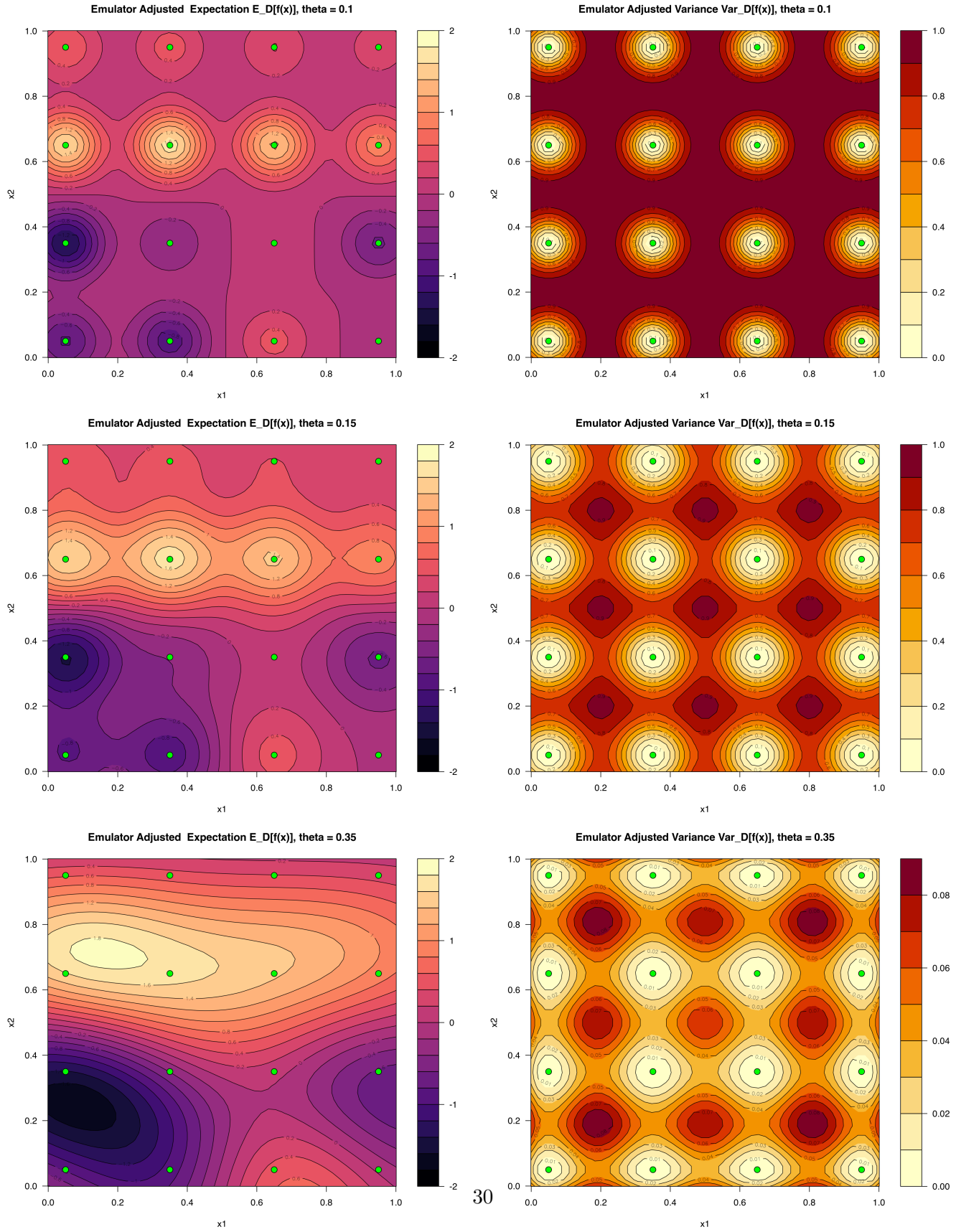
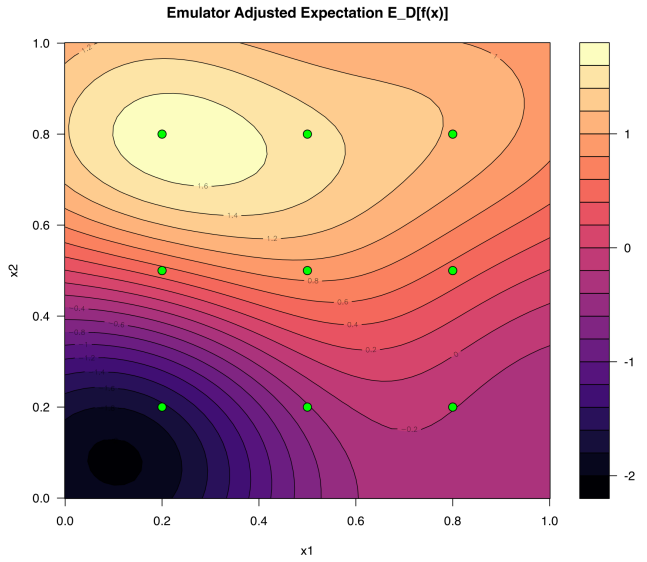


Figure 9: Adjusted expectation $\mathrm{E}_D[f(x)]$ (left column) and adjusted variance $\mathrm{Var}_D[f(x)]$ (right column) for differing values of $\theta = 0.1, 0.15$ and 0.35 (top, middle and bottom rows respectively). Note the vastly smaller maximum $\mathrm{Var}_D[f(x)]$ in the $\theta = 0.35$ case.

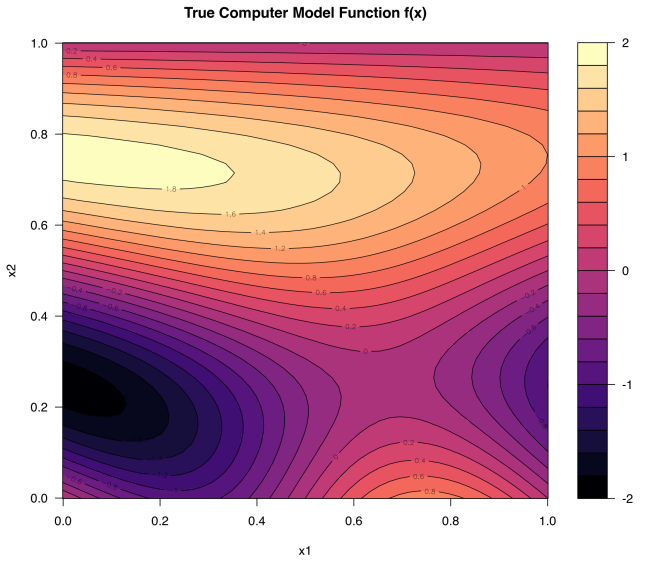
2.5 Designing Computer Model Experiments: Part I

- We used 16 runs to create a good emulator, but could we have used fewer?
- Figure 10 shows an emulator with the same θ and σ parameters, built with a grid of 9 runs formed from $\{0.2, 0.5, 0.8\} \times \{0.2, 0.5, 0.8\}$. Although it satisfies emulator diagnostics (shown in figure 10d), it is far less successful at mimicking $f(x)$.
- Perhaps the grid is too compressed. Figures 11 and 12 show larger 3×3 grids, however, the emulators constructed are again not satisfactory and actually violate emulator diagnostics as shown by the white areas of the corresponding diagnostic plots, implying $|S_D[f(x)]| > 3$ in those places (refer to the colour scale).
- Figure 13 uses a 5×5 grid of 25 runs and shows an accurate emulator, but not much more accurate that the original 16 point design, so the extra runs are perhaps not worth it.
- This highlights classic *Design of Experiments* questions:
  - How many runs do we need for sufficient accuracy?
  - Given a budget of n runs, where should we place them?
- We see that a good design has runs well spread out (i.e. no large holes).
- So is a grid (i.e. a full factorial) design good?
- Horrific in higher dimensions as scales terribly: e.g. $3 \times 3 \times \ldots$ grid in 10D requires $3^{10} = 59049$ runs and in 20D requires $3^{20} = 3.48$ billion runs: extremely expensive!
- Generally grid designs are very wasteful as each run is very similar to many other runs, just with a single input changed by a modest amount. This means we are repeating very similar calculations again and again. More efficient to change lots of input values to explore more.
- Grids are not bad in 2D though, depending on the function, but some runs wasted e.g. on edges and corners.
- Even in 2D there can be problems: the emulator in figure 12, which has runs on the edges and corners is pretty disastrous and doesn't mimic the true function at all. It has been fooled by the periodic nature of $f(x)$ which equals zero at certain points (see if you can work out where).
- The wasted repeatability issue of grid designs can be seen most clearly if some inputs are less active than others e.g. what about the function:

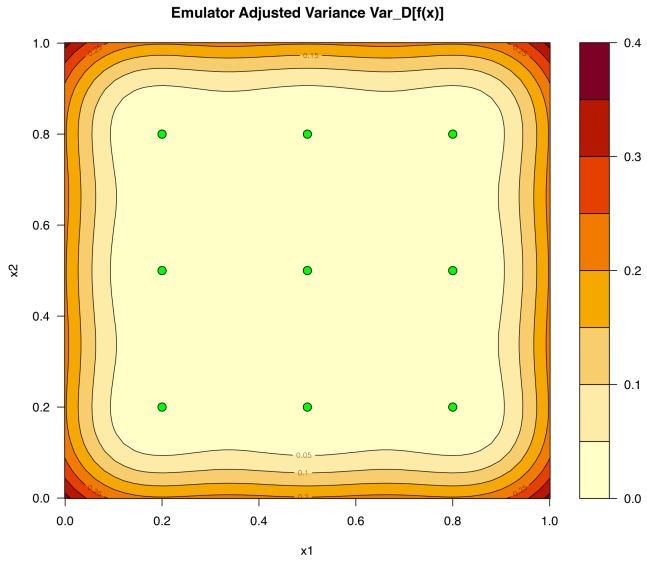
$$f(x) \;=\; f(x_1, x_2) \;=\; \sin(3\pi x_1) + \frac{1}{20}\cos(2\pi x_2) \tag{54}$$



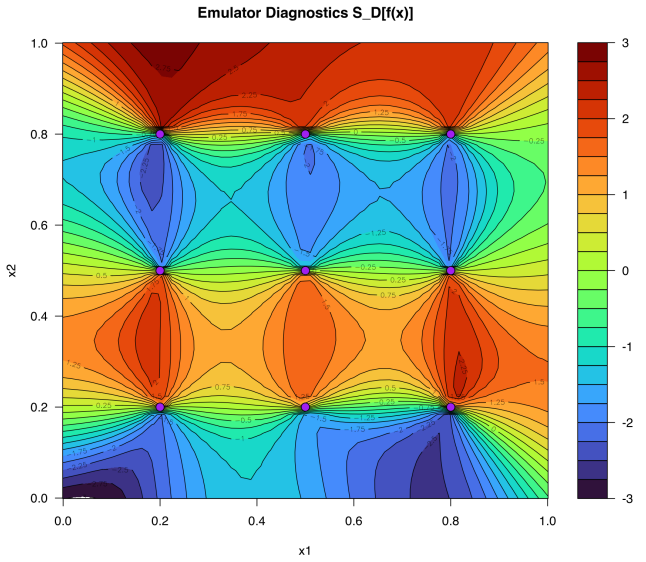
(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.



(b) True computer model function $f(x)$.

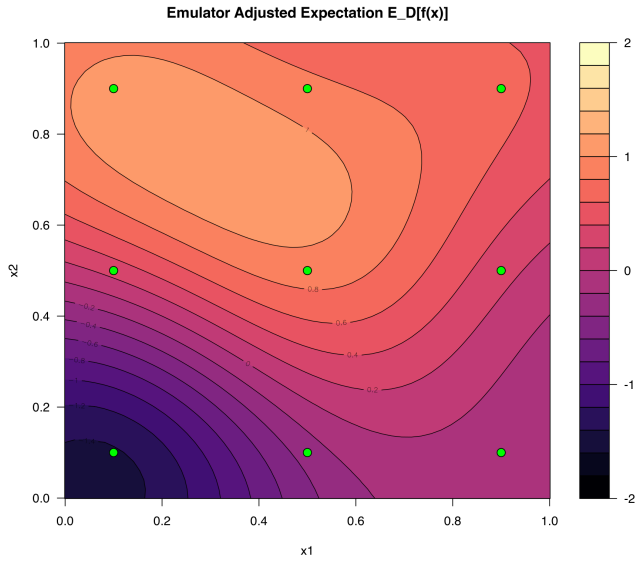


(c) Emulator Variance $\mathrm{Var}_D[f(x)]$.

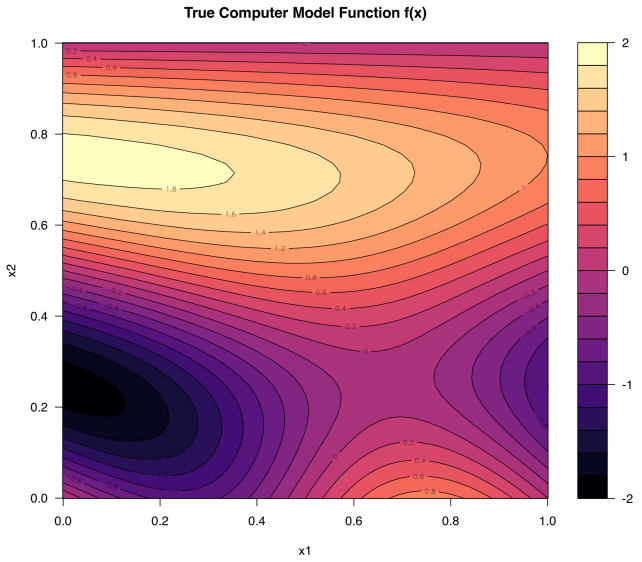


(d) Emulator diagnostics $S_D[f(x)]$.

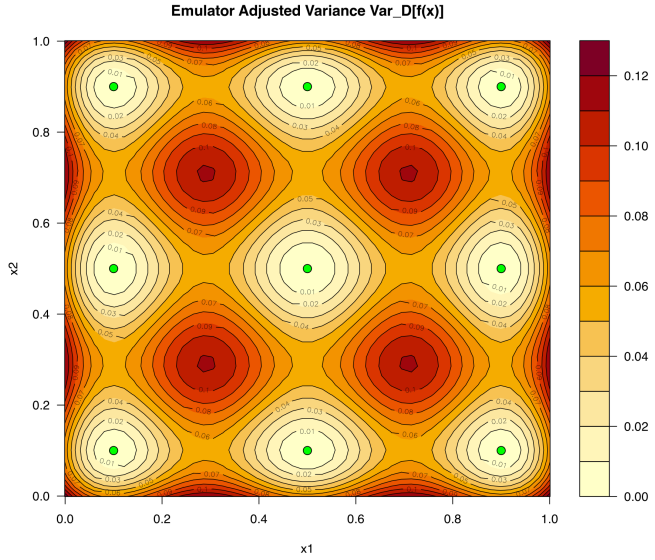
Figure 10: A two dimensional emulator using a compressed 9 point grid design. White regions in the diagnostic plot for $S_D[f(x)]$ (panel (d)) show areas where the emulator is failing diagnostics as $S_D[f(x)]$ is not within the interval $[-3, 3]$.



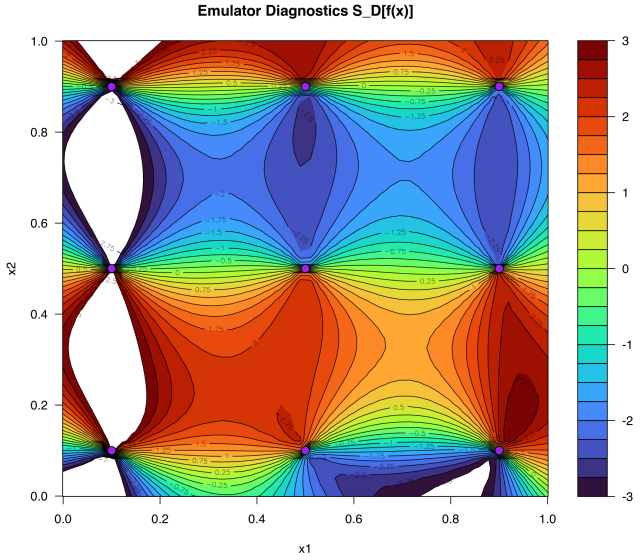
(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.



(b) True computer model function $f(x)$.

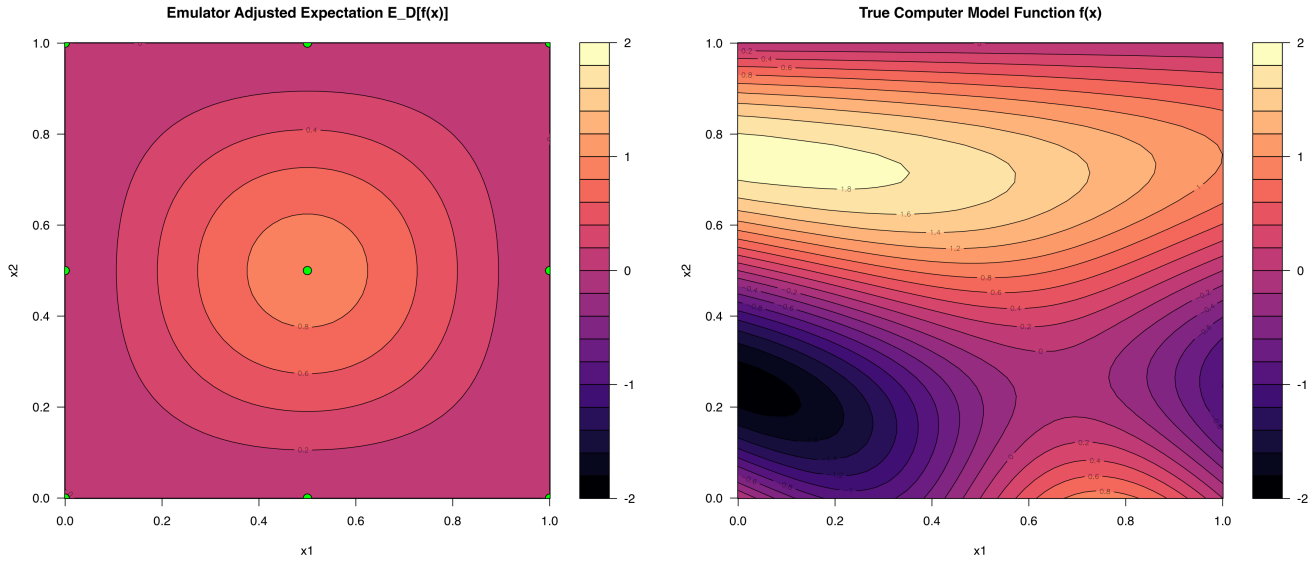


(c) Emulator Variance $\mathrm{Var}_D[f(x)]$.



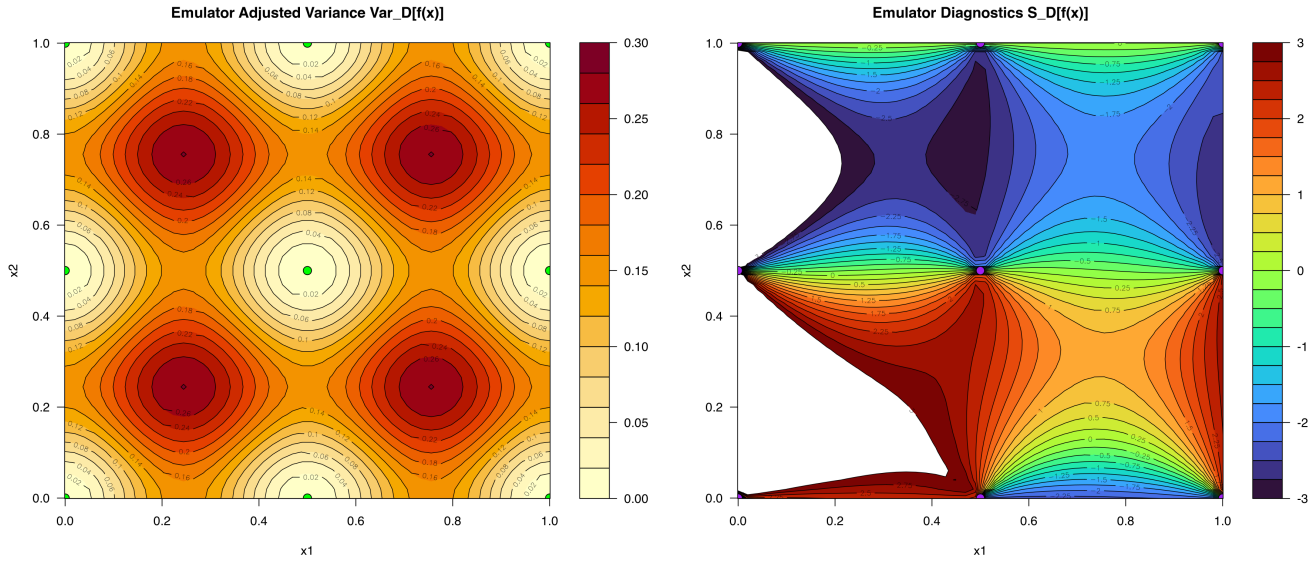
(d) Emulator diagnostics $S_D[f(x)]$.

Figure 11: A two dimensional emulator using a larger 9 point grid design. White regions in the diagnostic plot for $S_D[f(x)]$ (panel (d)) show areas where the emulator is failing diagnostics as $S_D[f(x)]$ is not within the interval $[-3, 3]$.



(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.

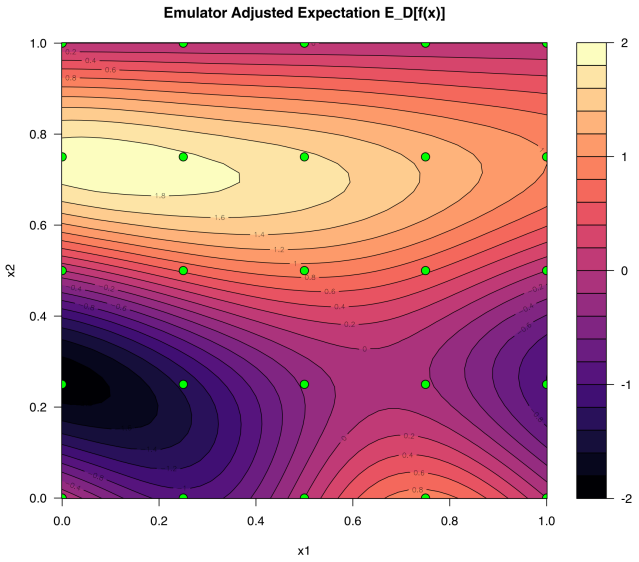
(b) True computer model function $f(x)$.



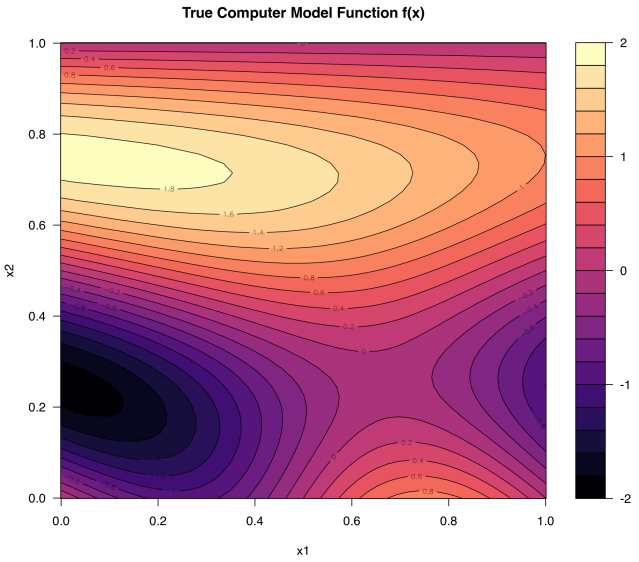
(c) Emulator Variance $\mathrm{Var}_D[f(x)]$.

(d) Emulator diagnostics $S_D[f(x)]$.

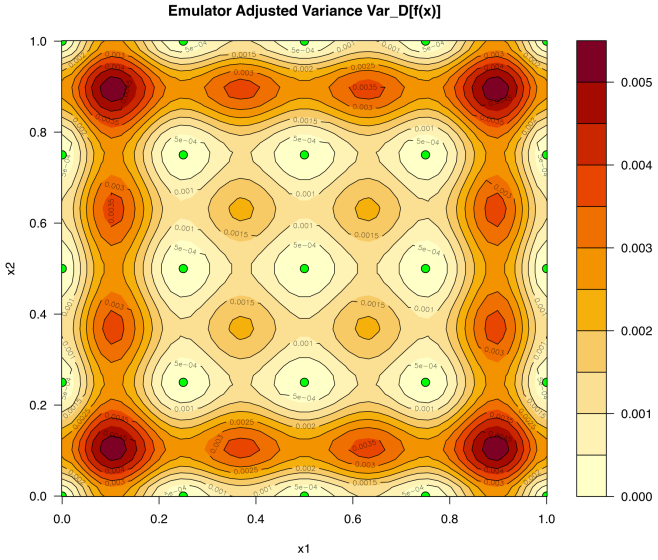
Figure 12: A two dimensional emulator using the largest 9 point grid design. White regions in the diagnostic plot for $S_D[f(x)]$ (panel (d)) show areas where the emulator is failing diagnostics as $S_D[f(x)]$ is not within the interval $[-3, 3]$.



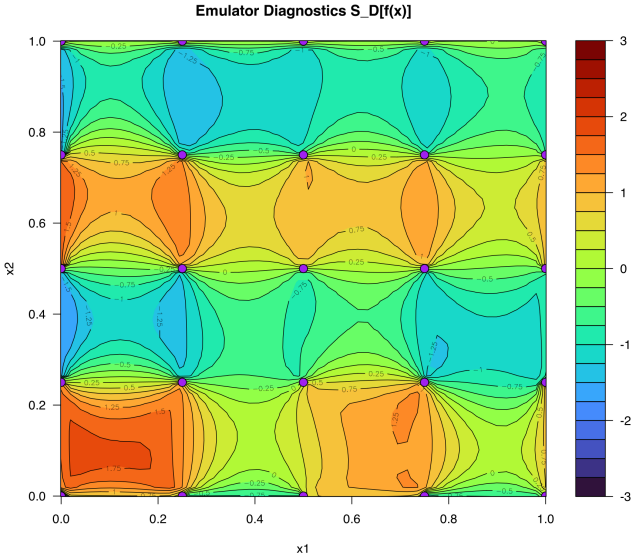
(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.



(b) True computer model function $f(x)$.



(c) Emulator Variance $\mathrm{Var}_D[f(x)]$.



(d) Emulator diagnostics $S_D[f(x)]$.

Figure 13: A two dimensional emulator using a 25 point grid design. White regions in the diagnostic plot for $S_D[f(x)]$ (panel (d)) show areas where the emulator is failing diagnostics as $S_D[f(x)]$ is not within the interval $[-3, 3]$.

- Note that x_2 now has a diminished effect on $f(x)$. We will discuss this issue in more detail later under the topic of "Active" and "Inactive" inputs, once we have introduced more advanced emulators, but note that it is very common in realistic computer models.
- We can see that a 4×4 grid design is very wasteful here as each set of 4 runs contain very similar information, as shown in figure 14.
- This implies a more randomised design may avoid such traps.
- However, a completely random design e.g. drawing each 2D run location $x^{(i)} \sim Unif[0,1]^2$ i.e. uniformly from the unit square $[0,1]^2$ is pretty bad too, as random points can have clumps of runs close together: again very wasteful.

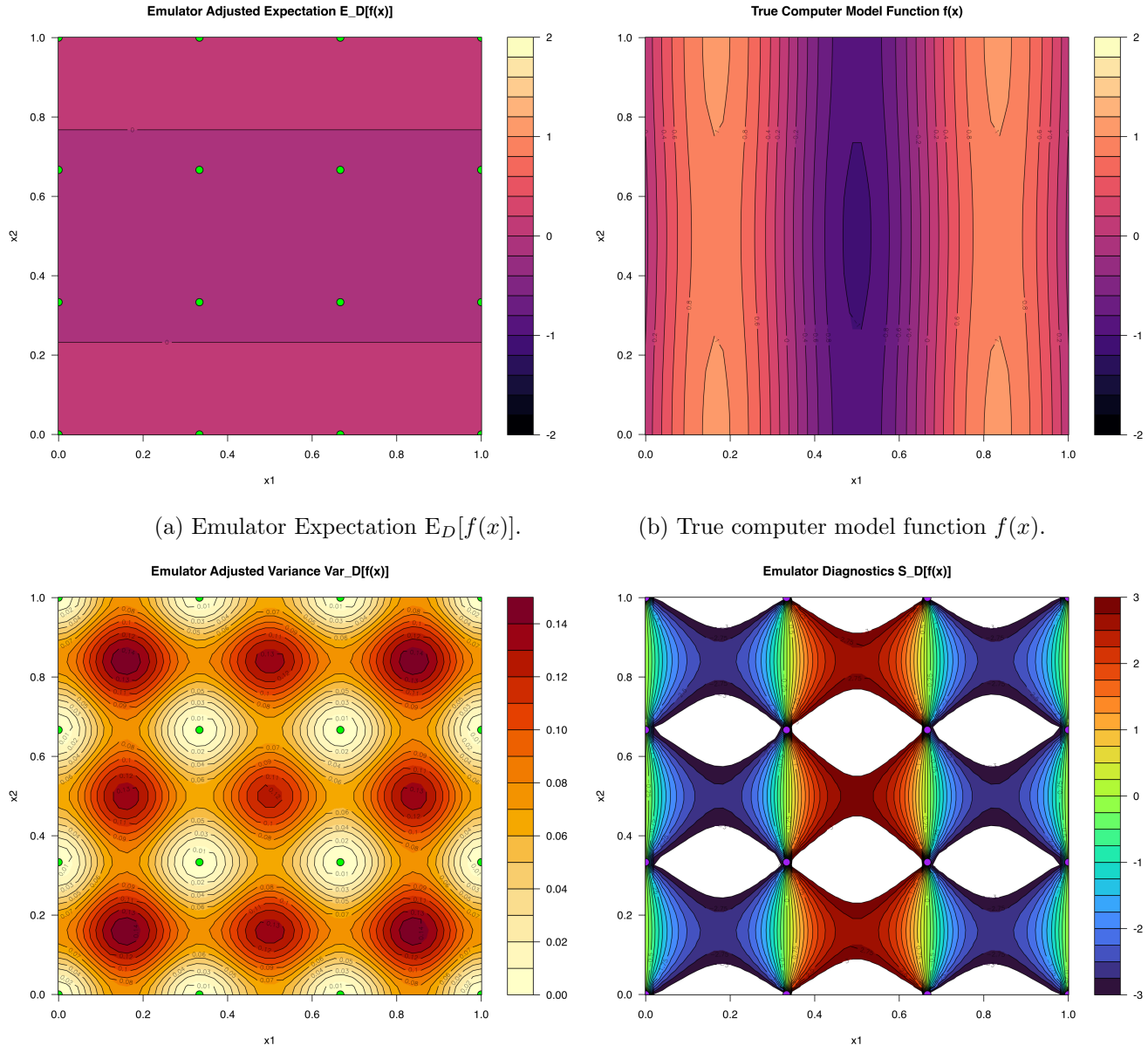
Latin Hypercube Designs

- The standard, generic design used in UQ is the Latin Hypercube Design (LHD).
- We will discuss their properties in more detail later, along with more advanced designs, but here we define them and highlight their main advantages.
- For m input dimensions, a Latin Hypercube design for n runs, is formed by
  - Dividing up the range of each input x_j into n equally sized intervals.
  - Placing the n runs so that each of the n intervals, for each of the inputs x_j, has exactly one run in it.
- This sounds hard, but is actually very easy. E.g. if the range for each input is $[0,1]$ then for each input x_j we choose a permutation of $\{0, 1/n, 2/n, \ldots, (n-1)/n\}$ and assign this to the jth input for each of the n runs: $x_j^{(k)}$, for $k = 1, \ldots, n$.
- We do this in turn for each of the m dimensions of the input space i.e. repeat for x_j with $j = 1, \ldots, m$.
- Then we either do:

Type 1: add $1/(2n)$ to each $x_j^{(k)}$ entry to recenter the points in the middle of the intervals $\{[0, 1/n), [1/n, 2/n), \ldots, [(n-1)/n, 1)\}$, or

Type 2: add random i.i.d. uniform draws $u_{jk} \sim U[0, 1/n]$ to each entry $x_j^{(k)}$ to randomly sample each point from its chosen m dimensional "box".

- Type 2 is considered beneficial for several reasons, e.g. the added randomness sidesteps any traps due to periodicity of the function.



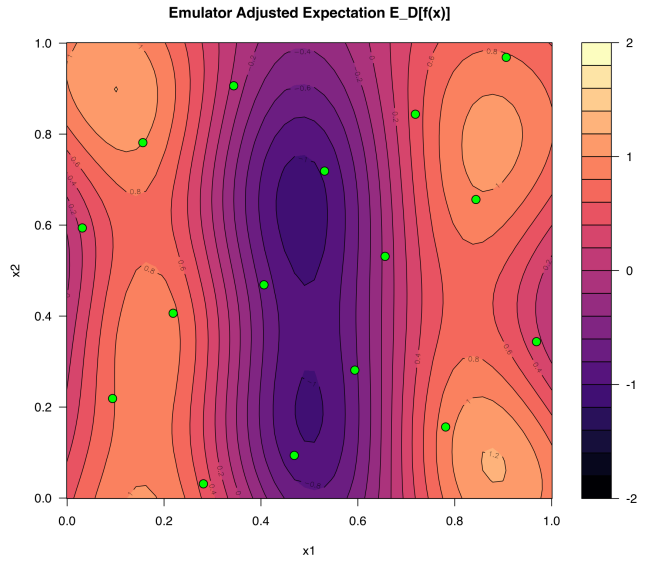
(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.

(b) True computer model function $f(x)$.

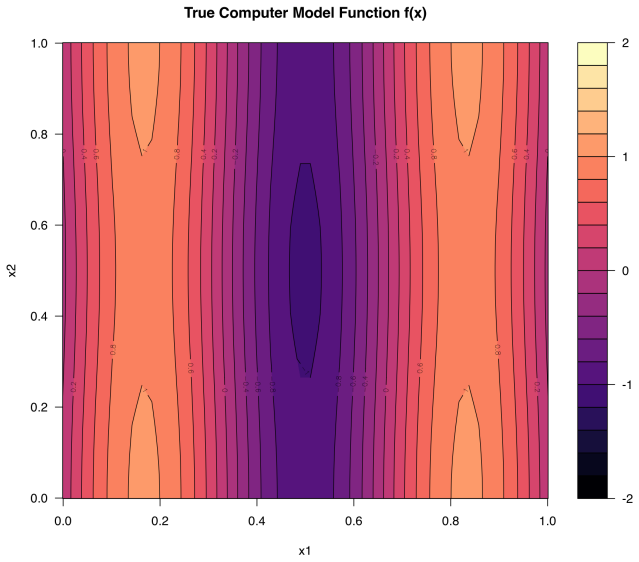
(c) Emulator Variance $\mathrm{Var}_D[f(x)]$.

(d) Emulator diagnostics $S_D[f(x)]$.

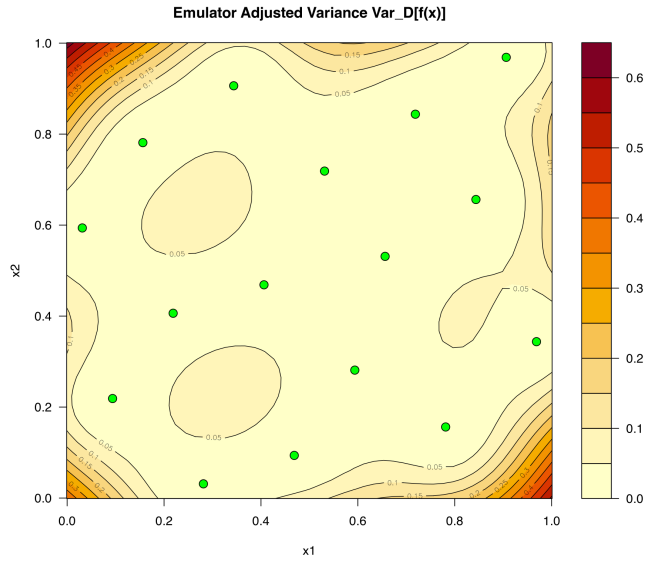
Figure 14: Example demonstrating the wasteful nature of a 4×4 grid design. The true function $f(x)$ is as given in equation (54) and shown in panel (b). As the x_1 input has a dramatic effect on $f(x)$ but the x_2 input only has a mild effect, the grid design wastes points as sets of 4 runs have the same x_1 value and hence very similar information. We are sampling x_1 at only 4 different values and hence can't capture its effects.



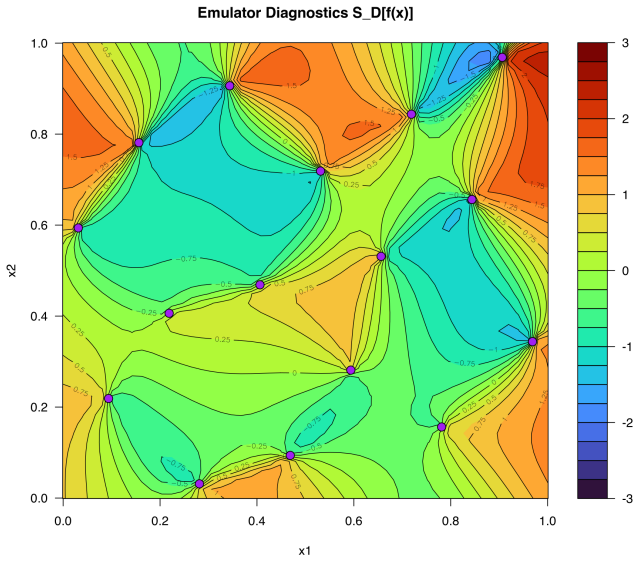
(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.



(b) True computer model function $f(x)$.



(c) Emulator Variance $\mathrm{Var}_D[f(x)]$.



(d) Emulator diagnostics $S_D[f(x)]$.

Figure 15: Example demonstrating the benefit of a maximin Latin Hypercube Design. The true function $f(x)$ is as given in equation (54) and shown in panel (b). As the Latin Hypercube Design samples uniformly over the range of x_1, ensuring there is a run in each of the 16 subintervals, it partially captures the strong x_1 behaviour (and a little of the subdominant x_2 behaviour). Diagnostics now look fine unlike in figure 14.

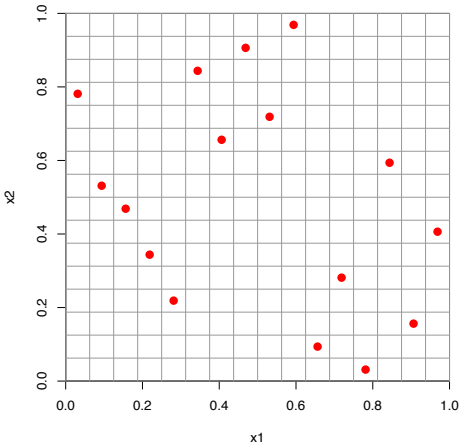
- The above construction guarantees marginal uniformity i.e. the design is pretty uniform along each input. However, this can still lead to poor joint designs when we look at more than one input at a time e.g. large holes in higher dimensional space.
- So often we optimise the LHD to have additional properties. The most common choice is the *Maximin LHD* where we find a LHD with the maximum minimum distance between two runs i.e. ensure no runs are close.
- Example Latin Hypercube Designs and Maximin Latin Hypercube Designs (with and without uniform draws) are shown in figure 16. See the Computer Practicals for more details on how to construct these.
- If we use a Maximin Latin Hypercube Design to emulate the problematic function given above in equation (54):

$$f(x) \quad = \quad f(x_1, x_2) \quad = \quad \sin(3\pi x_1) + \frac{1}{20}\cos(2\pi x_2) \tag{55}$$

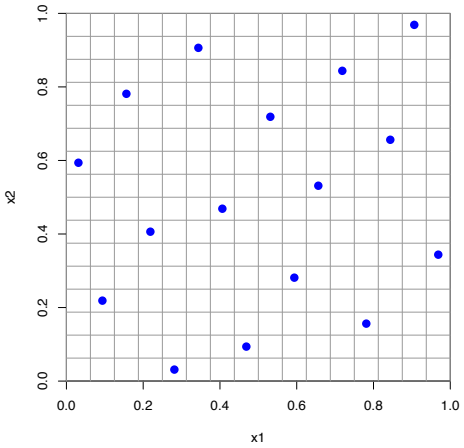
  we get significantly improved results as shown in figure 15, using $\theta = 0.35$ and $\sigma = 1$.
- As the Latin Hypercube Design samples uniformly over the range of x_1, ensuring there is a run in each of the 16 subintervals, it partially captures the strong x_1 behaviour (and captures a little of the subdominant x_2 behaviour).
- The emulator diagnostics now look fine, unlike in figure 14.
- We may return to more detailed design issues later, once we have developed more advanced emulators.

2.6 Covariance Structures: Part 1

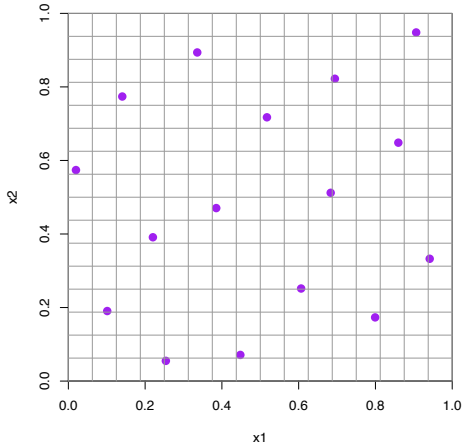
- The above simple emulators work because we have chosen a suitable covariance structure (see e.g. equation (52)), but how should we choose this?
- We often represent our prior beliefs using *weakly stationary processes* where the covariance structure depends on the difference between input points $x - x'$ and not on the absolute location in input space (this is what the word "stationary" implies, while "weakly" implies this is true for the first and second order quantities).
- We often go further and assume isotropy so that the covariance structure depends on the distance between points $r = ||x - x'||$ and hence not on the direction.
- Note that this means our *prior beliefs about* $f(x)$ are stationary and isotropic, not that we think $f(x)$ is!



(a) Latin Hypercube Design.



(b) Maximin Latin Hypercube Design.



(c) Maximin LHD with uniform draws.

Figure 16: Three $n = 16$ Latin Hypercube designs. (a) and (b) are Type 1 (points in centre of boxes) while (c) is Type 2 (points uniformly drawn from within local box).

- So we have:

$$\text{Cov}[f(x), f(x')] \;=\; \sigma^2 c(x, x') \;=\; \sigma^2 c(x - x') \;=\; \sigma^2 c(||x - x'||) \qquad (56)$$

  where $c(x, x')$ is often called the *kernal*. We shall see this stationary, isotropic form in the following popular structures.

- Note that the stationarity is now clear as the covariance is invariant to translations:

$$\text{Cov}[f(x+a), f(x'+a)] \;=\; \sigma^2 c((x+a)-(x'+a)) \;=\; \sigma^2 c(x-x') \;=\; \text{Cov}[f(x), f(x')]$$

- Various choices of covariance structure (or equivalently kernal) exist, and we choose them according to our prior beliefs about the function $f(x)$.

The Squared Exponential Covariance Function

$$\mathrm{Cov}[f(x), f(x')] \;=\; \sigma^2 \exp\left\{-\frac{||x-x'||^2}{\theta^2}\right\} \tag{57}$$

- The classic which we have used so far. As mentioned above, this is good if we think $f(x)$ is smooth i.e. infinitely differentiable.
- Can be too smooth and lead to poor emulator diagnostics.

The Matérn Covariance Function

- If we believe that say k derivatives of $f(x)$ exist (and no more) then a good covariance function is the Matérn: which is given by:

$$\mathrm{Cov}[f(x), f(x')] \;=\; \sigma^2\, \frac{2^{1-\nu}}{\Gamma(\nu)} \left(2\sqrt{\nu}\frac{||x-x'||}{\theta}\right)^{\nu} K_\nu \left(2\sqrt{\nu}\frac{||x-x'||}{\theta}\right), \tag{58}$$

  where K_ν is a modified Bessel function of the third kind and θ is the usual correlation length but now ν is a parameter that governs the number of derivatives of the computer model that exist. Specifically, ν rounded down to the next integer gives the number of derivatives k that exist (see below for details).

- The Matérn covariance function has the useful property that in the $\nu \to \infty$ limit it tends to the Squared Exponential covariance given in equation (57).
- When ν is half integer, the Matérn function simplifies to an exponential times a polynomial, and the most used versions are for $v = 3/2$ and $\nu = 5/2$ which give:

$$v = 3/2, \qquad \mathrm{Cov}[f(x), f(x')] \;=\; \left(1 + \frac{\sqrt{6}r}{\theta}\right) \exp\left\{-\frac{\sqrt{6}r}{\theta}\right\} \tag{59}$$

$$v = 5/2, \qquad \mathrm{Cov}[f(x), f(x')] \;=\; \left(1 + \frac{\sqrt{10}r}{\theta} + \frac{10r^2}{3\theta^2}\right) \exp\left\{-\frac{\sqrt{10}r}{\theta}\right\}$$

$$\text{with} \qquad r = ||x-x'||$$

The Exponential Covariance Function

- If we don't think that $f(x)$ is even differentiable we can use the exponential covariance function:

$$\mathrm{Cov}[f(x), f(x')] \;=\; \sigma^2 \exp\left\{-\frac{||x-x'||}{\theta}\right\} \tag{60}$$

  which is actually a special case of the Matérn with $\nu = 1/2$ and θ rescaled to $\sqrt{2}\,\theta$.

- This is useful if we think $f(x)$ is continuous but not differentiable.
- We will revisit covariance functions and their effects when we start to simulate realisations from the emulator which will highlight the impact of different covariance choices.

Numerical Stability

- The above is all fine, except that we implement emulators on computers which possess finite precision.
- In some cases, covariance functions, especially the Squared Exponential, can lead to variance matrices that are ill-conditioned and which when combined with finite numerical precision lead to matrices (e.g. Var$[D]$) that are not positive semi-definite (i.e. have small negative eigenvalues, and hence are not valid covariance matrices).
- An easy fix is to include a small *nugget term* which introduces a small amount of noise (to account for the numerical imprecision) on top of an otherwise smooth process. This changes say the Squared Exponential to

$$\text{Cov}[f(x), f(x')] \;=\; \sigma^2 \left[\exp\left\{ -\frac{||x - x'||^2}{\theta^2} \right\} \;+\; \delta \, \mathbb{1}_{x=x'} \right] \tag{61}$$

  where $\mathbb{1}_A$ is the indicator function that takes value 1 when statement A is true and 0 otherwise, and δ is a small value often say $\delta = 10^{-6}$, which governs the size of the additional noise as a proportion of the variance σ^2.
- For our emulator construction, this just adds $\sigma^2\delta$ to the diagonal of the Var$[D]$ matrix. We will discuss more about nuggets (as they have two other important uses) when we discuss more complex emulator structures.

Different Correlation Lengths θ_i for Different Inputs x_i

- In all the above covariance function, the correlation length θ acts on each input dimension equally, rescaling distances between points.
- However, we may wish to break the isotropy and rescale distances differently along each input dimension for cases such as shown in figure 14 and 15.
- We can do this by generalising to individual correlation lengths θ_i, $i = 1, \ldots, m$ corresponding to each input x_i.

- For example, in m input dimensions the Squared Exponential Covariance Function generalises to:

$$\text{Cov}[f(x), f(x')] \;=\; \sigma^2 \exp\left\{ -\sum_{i=1}^{m} \frac{(x_i - x_i')^2}{\theta_i^2} \right\} \tag{62}$$

- This reduces to the previous isotropic version if we set $\theta_i = \theta$, $\forall i = 1, \ldots, m$.

- An example of the use of this anisotropic covariance structure is shown in figure 17 where we have chosen $\theta_1 = 0.35$ and $\theta_2 = 1.4$, with $\sigma = 1$ as before. This helps to capture both the more dramatic changes in $f(x)$ caused by x_1 and the more subtle changes caused by x_2. Compare with the standard isotropic version in figure 15.

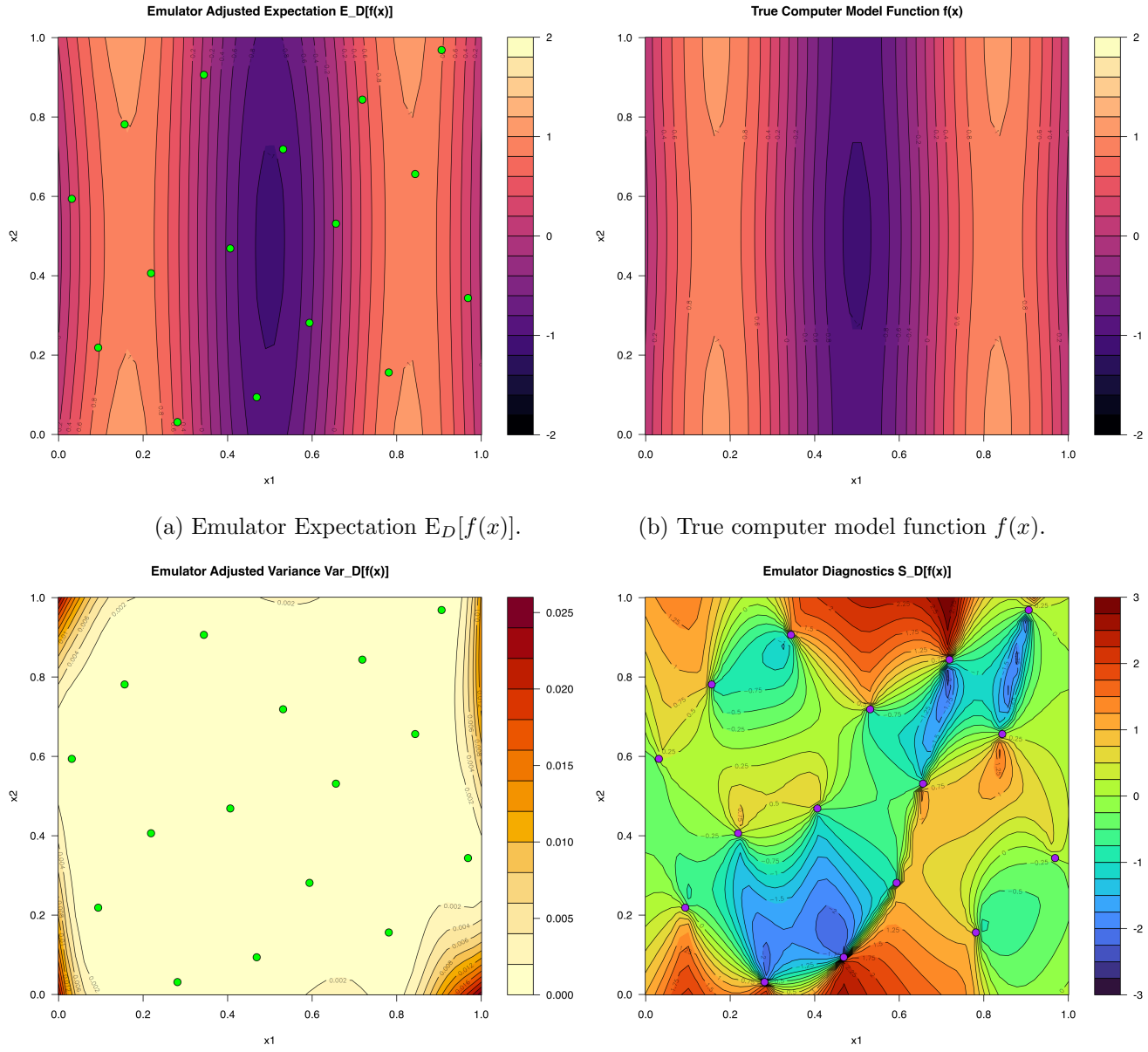
- Figure 18 shows the emulator expectation and variance for this same anisotropic case, but now with varying levels of θ_1 and θ_2.

- Note that for lower θ_i values e.g. as shown in the top row, we can see that the runs now have an ellipsoidal influence around them instead of the circular influence seen before in the isotropic case of figure 9.

Technical aside: Mean Square Continuity and Differentiability

- Technically, when we discuss the differentiability of processes we need to be a little careful.

- The above covariance functions ensure the continuity or differentiability of the process in the mean square sense, which is slightly different from the realisations themselves, but enough for our needs. See e.g. Rasmussen and Williams for further details.



(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.

(b) True computer model function $f(x)$.

(c) Emulator Variance $\mathrm{Var}_D[f(x)]$.

(d) Emulator diagnostics $S_D[f(x)]$.

Figure 17: Example demonstrating the benefit of an anisotropic covariance structure as given by equation (62). The true function $f(x)$ is as given in equation (54) and shown in panel (b). Here $\theta_1 = 0.35$ while $\theta_2 = 1.4$ which helps to capture both the more dramatic changes in $f(x)$ caused by x_1 and the more subtle changes caused by x_2. Compare with the standard isotropic version in figure 15.

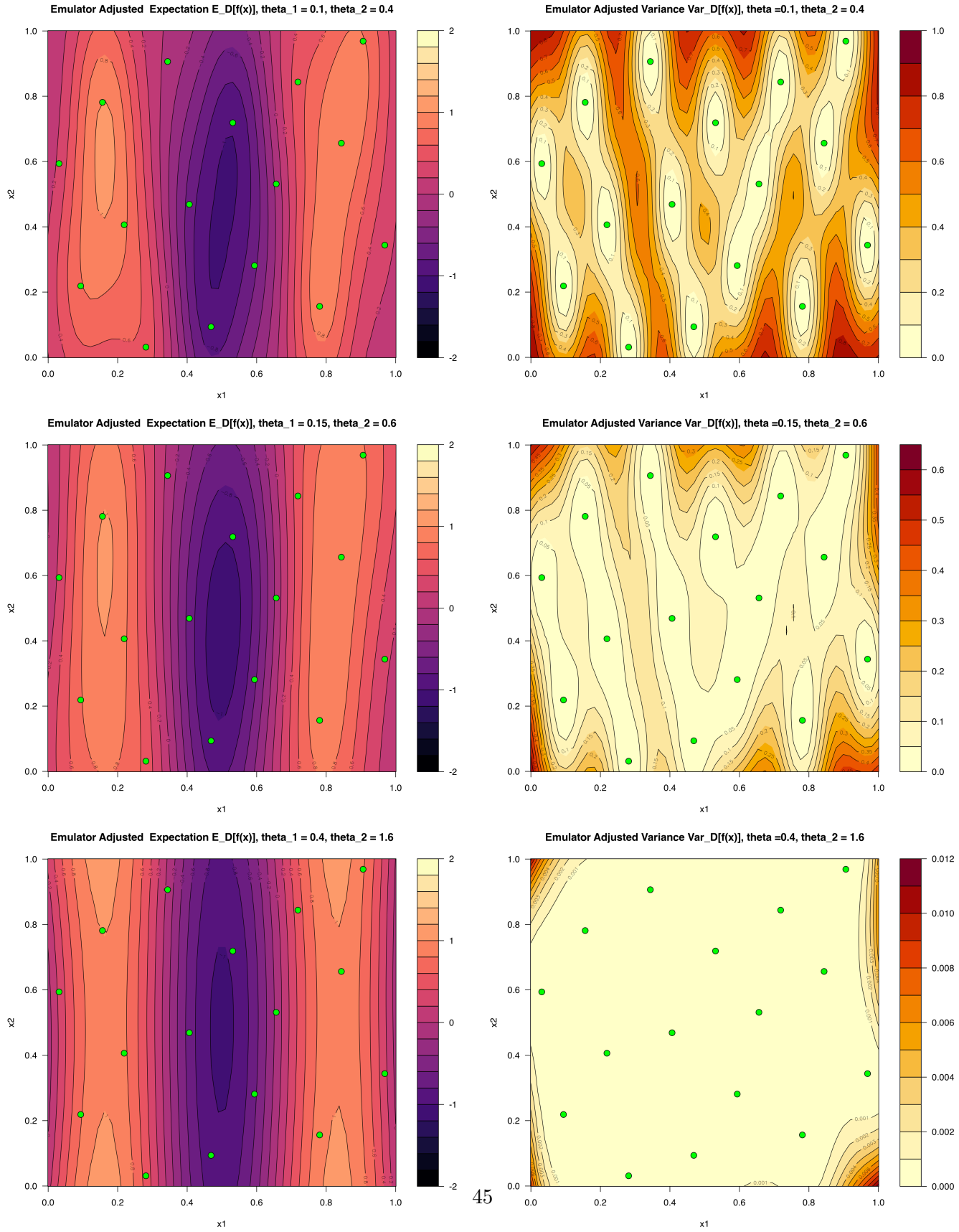


Figure 18: Adjusted expectation $\mathrm{E}_D[f(x)]$ (left column) and adjusted variance $\mathrm{Var}_D[f(x)]$ (right column) for differing values of θ_1 and θ_2 (top, middle and bottom rows) for the anisotropic covariance structure of equation (62). Note the ellipsoidal impact of the runs.

3 Linking to Reality

The Real System y

- Emulators are great for mimicking the expensive function $f(x)$ and examining and visualising its behaviour.
- But why do we care?
- Usually, scientists employ a complex model $f(x)$ to mimic some real world system of importance (where f may now give scalar or vector output). E.g. climate models are used to mimic Earth's real climate, galaxy formation models are used to mimic the type of galaxies that formed in our Universe etc.
- x typically represents a list of physical properties that we are uncertain about.
- Critically, often scientists forget that"

  "The model $f(x)$ is NOT THE SAME AS THE REAL SYSTEM."

- We denote the real system output as y: but we rarely ever see this.

The Model Discrepancy ϵ

- Now, even were we to pick the best possible input $x = x^*$ to our model $f(x)$, to make it best agree with the system, it would still not be in precise agreement with the system y due to the many simplifications and approximations of the model (both in terms of the model and its implementation on a computer).
- We represent this discrepancy between model and real system using a simple statistical model (we can make this more complex if needed):

$$y \;=\; f(x^*) \;+\; \epsilon \tag{63}$$

  where ϵ is a random quantity that represents the *Model Discrepancy* i.e. the difference between the model evaluated at its best (or at least acceptable) input x^* and the real system we are trying to mimic. We typically assume/assert that ϵ is independent of $f(x^*)$.
- We may express our prior beliefs about ϵ via second order statements: $\mathrm{E}[\epsilon]$ and $\mathrm{Var}[\epsilon]$ in a Bayes Linear analysis, or via full probabilistic specification if needed in a full Bayesian analysis. Or we can learn about ϵ in either paradigm.
- Including the Model Discrepancy and the uncertainty it represents is critical: without it, any subsequent analysis assumes that the model $f(x)$ is a *perfect mimic of the system for some value of* x which is rarely, if ever, true.

- Assuming the model is perfect in this way will poison subsequent inference about x (we become overconfident), poison any future predictions (we become doubly overconfident!) and therefore damage any decision support (decisions become non-robust and untrustworthy).

But this is a mistake that the majority of scientific groups make!

Observations z of the Real System

- Although we rarely know the real system y we can perform imperfect observations of it denoted z, usually performed with some observational error. We represent the relationship between system and imperfect observations using the simple statistical model (again which can be made more complex if necessary):

$$z \;=\; y \;+\; e \tag{64}$$

  where z are the imperfect observations and e is a random quantity that represents the unknown observational errors. We again may express our prior beliefs about e in terms of $\mathrm{E}[e]$ and $\mathrm{Var}[e]$, based on our knowledge of the observation process, or again give a fully probabilistic specification.

- The observation error e is far more familiar (as scientists are used to imperfect observations) and so is typically included by most scientific groups, albeit in often simple and naive forms.

Specifying Model Discrepancy and Observational Error Structures

- Often we may specify $\mathrm{E}[e] = 0$ and $\mathrm{E}[\epsilon] = 0$, if we think there are no obvious biases etc. The real impact, however, is often due to the $\mathrm{Var}[e]$ and $\mathrm{Var}[\epsilon]$ terms.

- Note that if $f(x)$ provides vector output of length q say, then $\mathrm{Var}[e]$ and $\mathrm{Var}[\epsilon]$ will both be $q \times q$ covariance matrices.

- Say $f(x)$ represented the global mean temperature each week for a ten week period, so $q = 10$.

- We may view that the observation errors each week were uncorrelated (just to do with scientific equipment/satelite measurement error etc.) and hence would assert that $\mathrm{Cov}[e_i, e_j] = 0$ for $i \neq j$ and hence $\mathrm{Var}[e] = \mathrm{diag}\{\sigma^2_{e_1}, \sigma^2_{e_2}, \ldots, \sigma^2_{e_{10}}\}$.

- However, if we suspect the model $f(x)$ will over or under shoot the real mean temperature *for the whole 10 week period* we would assert high correlation between elements of ϵ so that $\mathrm{Cor}[\epsilon_i, \epsilon_j]$ is large (and very large for $j = i \pm 1$ i.e. adjacent weeks).

Summary

- We actually care about the real system y. The model $f(x)$ is just a tool which informs about y, approximating how system properties affect system behaviour.
- The nature of the covariance matrices $\text{Var}[e]$ and $\text{Var}[\epsilon]$ are very different, both in size and correlation structure. Ignoring either will tragically undermine subsequent analysis.
- There are a set of techniques for specifying Model Discrepancy (as this may seem a daunting process) which break it into two pieces: the internal discrepancy (which can be assessed from further runs of a perturbed version of the model) and external discrepancy (which contains everything else that is missing).
- Model Discrepancy is very important when we try to learn about x^* (e.g. using History Matching, which we will do in the next section) and when we wish to do forecasting into the future, as the interplay between the MD in the past and in the future will become important.
- The inclusion of Model Discrepancy elevates our analysis to be an analysis of the real system (albeit with appropriate additional uncertainties) instead of an analysis of the model.
- Lets be very clear about the dangers of ignoring many sources of uncertainty. In addition to turning real science into just 'playing with models', it leads to:
  - **Overconfident inference** regarding physical parameters and processes.
  - **Doubly overconfident predictions/forecasts** of unmeasured/future outputs.
  - **Defective decision making** and **design of experiments**.
  - **Misleading model development**.
  - Overly **difficult optimisation** of decision criteria, chasing **spurious solutions** to **badly defined objective functions**.
  - **People dying...**

4 History Matching

4.1 Learning about Acceptable Input Parameters

- Reminder:
  - x is typically a list of (unknown) physical parameters representing properties of the system e.g. disease transmission rates in a disease model, defined over an initial input space $\mathcal{X}_0$ such that $x \in \mathcal{X}_0 \subset \mathbb{R}^m$.
  - $f(x)$ is a list of q outputs of the model of interest e.g. number of people with disease at various time points.
- Often a major goal is to use the computer model $f(x)$ along with observed data z of the real system y to learn about acceptable inputs x. This prompts the questions:
  1. Are there any input parameter settings x that lead to acceptable matches between the model output $f(x)$ and observed data z?
  2. If so, what is the full set $\mathcal{X}$ that contains all such input parameter settings?
- If the computer model $f(x)$ is slow, and the vector x moderate to high-dimensional, this is an incredibly challenging task.
- But it is vital as it is often the major scientific goal and is critical for several further calculations e.g. forecasting, decision support, designing future experiments etc.
- **History Matching** is a powerful iterative global parameter search method that exploits emulation to efficiently explore $\mathcal{X}_0$ to find the set of acceptable inputs $\mathcal{X}$.
- History Matching has been successfully employed in a variety of scientific disciplines including galaxy formation [1, 2, 3, 4, 5], epidemiology [6, 7, 8, 9, 10, 11], climate modelling [12, 13, 14, 15], oil reservoir modelling [16, 17, 18, 19], systems biology [20, 21], environmental science [22, 23], traffic modelling [24], nuclear physics [25, 26] and ice sheet modelling [27].
- Note that it is not a fully Bayesian procedure (e.g. we don't elicit priors over x, or more precisely x^* yet), but it can be an ideal precursor to full Bayesian Calibration/inference when this is deemed relevant (more on this later!).

4.2 Implausibility Measures (Univariate Output)

- History Matching employs *Implausibility Measures* to label parts of the initial input space $\mathcal{X}_0$ as worthy of further investigation.

- Assuming univariate (i.e. scalar) output $f(x)$ to start with, we can ask the question: for an unexplored input parameter setting x, how far would the emulator's expected value for the individual function output $f(x)$ be from the corresponding observed value z before we could deem it highly unlikely for $f(x)$ to give an acceptable match were we to evaluate the function at this value of x?

- The implausibility measure $I(x)$ captures this concept, and (its square) is given by:

$$\begin{aligned} I^2(x) &= \frac{(\mathrm{E}_D[f(x)] - z)^2}{\mathrm{Var}(\mathrm{E}_D[f(x)] - z)} \\ &= \frac{(\mathrm{E}_D[f(x)] - z)^2}{\mathrm{Var}_D[f(x)] + \mathrm{Var}[\epsilon] + \mathrm{Var}[e]} \end{aligned} \tag{65}$$

- The numerator of equation (65) gives the distance between the emulator expectation $\mathrm{E}_D(f(x))$ and the observation z, while the denominator standardises this quantity by all the relevant uncertainties regarding this distance (viewing z as uncertain but D as known), namely:
  - the emulator variance $\mathrm{Var}_D(f(x))$,
  - the model discrepancy variance $\mathrm{Var}(\epsilon)$, and,
  - the observation error variance $\mathrm{Var}(e)$.

- This structure is a direct consequence of the model discrepancy and observational error equations (63) and (64) along with assuming x is a candidate for x^*.

- **A large value of** $I(x)$ for a particular x occurs because the emulator expectation $\mathrm{E}_D[f(x)]$ is far from the observed data z, even given the uncertainties due to the emulator, the model discrepancy and the observation error.

- Therefore, we would be unlikely to obtain an acceptable match between $f(x)$ and z were we to run the model there.

- Hence we can discard the input x from the parameter search if $I(x) > c$, for some cutoff c, and refer to such an input as **implausible**.

- **A small value of** $I(x)$ can occur for two reasons: a) because the emulator expectation $\mathrm{E}_D[f(x)]$ is indeed close to the observed data z, or b) the uncertainties in the denominator of the implausibility, e.g. the emulator uncertainty $\mathrm{Var}_D[f(x)]$ are large.

- In this case we do not refer to to x as 'plausible' as after further investigation e.g. by performing more runs to reduce the emulator uncertainty, we may subsequently rule x out.

- We instead refer to such low-implausibility inputs as **non-implausible** with a deliberate use of a double negative.
- Note that such inputs are excellent places for further runs as they are likely to either yield acceptable matches or be highly informative in terms of their capability to reduce the emulator variance.
- We may choose the cutoff c by appealing to Pukelsheim's 3-sigma rule [28], which is the powerful result that states that for *any* continuous, unimodal distribution, 95% of its probability must lie within $\pm 3\sigma$ of its mean, regardless of asymmetry, skew, or heavy tails(!), which suggests that a choice of $c = 3$ could be deemed reasonable [1].

Example: Implausibility Measures in 1D

- Recall the 1D emulation example from section 2.2, of $f(x) = \sin(2\pi x)$. We will use this, however we will move the fifth design point to $x^{(5)} = 0.86$ to make the HM process clearer.
- Now imagine we have taken an imperfect observation of the real system y yielding value $z = -0.8$, with Observation Error (OE) variance $\text{Var}[e] = 0.01^2$ and Model Discrepancy (MD) variance $\text{Var}[\epsilon] = 0.02^2$.
- We can now construct the corresponding 1D Implausibility measure $I(x)$ (as there is only one output there is just one implausibility), for any x.
- E.g. for the point $x = 0.3$ we have (note the square root now on right hand side):

$$I(0.3) = \frac{|\text{E}_D[f(0.3)] - z|}{\sqrt{\text{Var}_D[f(0.3)] + \text{Var}[\epsilon] + \text{Var}[e]}} \quad (66)$$

$$= \frac{|0.98279 - (-0.8)|}{\sqrt{0.0052597 + 0.02^2 + 0.01^2}} \quad (67)$$

$$= 23.49 \qquad \text{(to 4 s.f.)} \quad (68)$$

  implying that $x = 0.3$ is highly implausible and unlikely to yield a good match.
- We can calculate $I(x)$ for a large set of 201 x values given by x_P as before and plot the result. This is shown in figure 19b. The x-axis is coloured to highlight the effect of the implausibility and the implausibility cutoff $I(x) < 3$ is shown as the green horizontal line.
- Compare with the emulator shown in figure 19a which also shows the observation z (horizontal black line) and interval $z \pm 3\sqrt{\text{Var}[e] + \text{Var}[\epsilon]}$ (horizontal black dashed lines).

- We see that $I(x)$ is very large at known run locations for runs not close to z: here we are sure $f(x)$ is a bad match to z. However $I(x)$ goes small when $\mathrm{E}[f(x)]$ is close to z and when $\sqrt{\mathrm{Var}[f(x)]}$ is large compared to $\mathrm{E}[f(x)] - z$, as expected.
- Examining the regions of high $I(x)$, we see that it would be wasteful to place further runs there if we are only interested in x values that will give a good match between $f(x)$ and z. Placing runs where $I(x)$ is low will either yield good runs or will reduce the emulator uncertainty in the non-implausible regions (or both!).

4.3 Implausibility Measures (Multivariate Output)

- If $f(x)$ gives multivariate output i.e. output dimension of f is q where $q > 1$, then there is a selection of measures we can use.
- We can calculate an implausibility measure $I_i(x)$ for each individual output $f_i(x)$ with $i = 1, \ldots, q$, as

$$I_i^2(x) \;=\; \frac{(\mathrm{E}_{D_i}[f_i(x)] - z_i)^2}{\mathrm{Var}_{D_i}[f_i(x)] + \mathrm{Var}[\epsilon_i] + \mathrm{Var}[e_i]} \tag{69}$$

  where $\mathrm{E}_{D_i}[f_i(x)]$ and $\mathrm{Var}_{D_i}[f_i(x)]$ are the individual emulators for output i etc.
- Then we can combine these individual implausibilities $I_i(x)$ together using maximisation, to define the maximised implausibility:

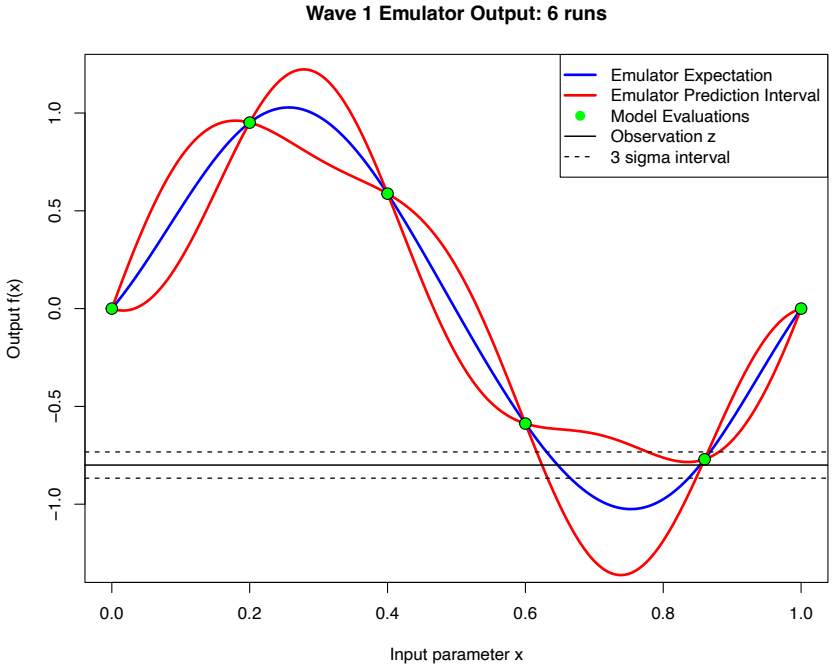
$$I_M(x) \;=\; \max_i I_i(x) \tag{70}$$

  Imposing the constraint $I_M(x) < c_M$ essentially insists that all outputs have individual implausibilities $I_i(x)$ less than c_M.
- While $I_M(x)$ is very useful, it is sensitive to any failures of individual emulators so we sometimes construct slightly more conservative implausibility measures by finding the second and third (or even nth) max implausibilities:

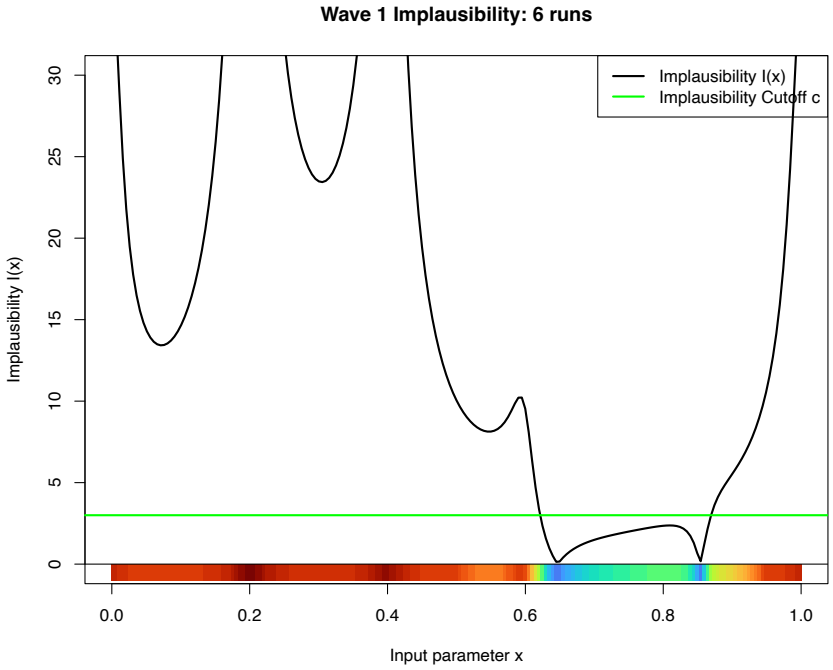
$$I_{2M}(x) \;=\; \max_i(\ \{I_{(i)}(x)\} \setminus I_M(x)\ ), \tag{71}$$

$$I_{3M}(x) \;=\; \max_i(\ \{I_{(i)}(x)\} \setminus \{I_M(x), I_{2M}(x)\}\ ), \tag{72}$$

  and use cutoffs c_{2M} and c_{3M} respectively. Imposing the cutoff $I_{2M}(x) < c_{2M}$ for example, insists that all outputs except one have individual implausibilities less than the cutoff of c_{2M}.
- We may impose several measures at once, and choose cautious values for c_M, c_{2M} and c_{3M} based on Pukelsheim's 3-sigma rule and considering how many outputs there are.

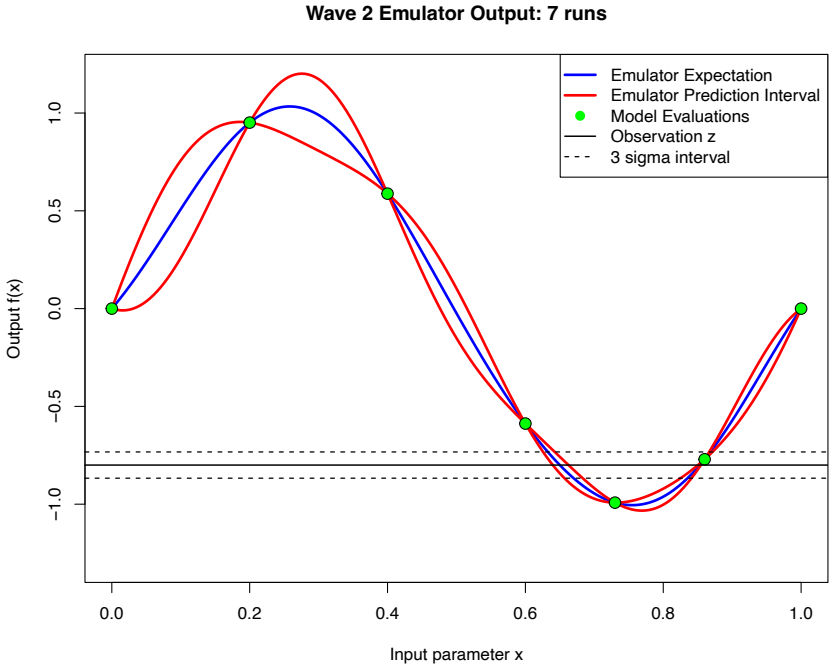


(a) Simple 1D emulator showing observed data $z \pm 3\sqrt{\sigma_e^2 + \sigma_\epsilon^2}$ for use in History Match.

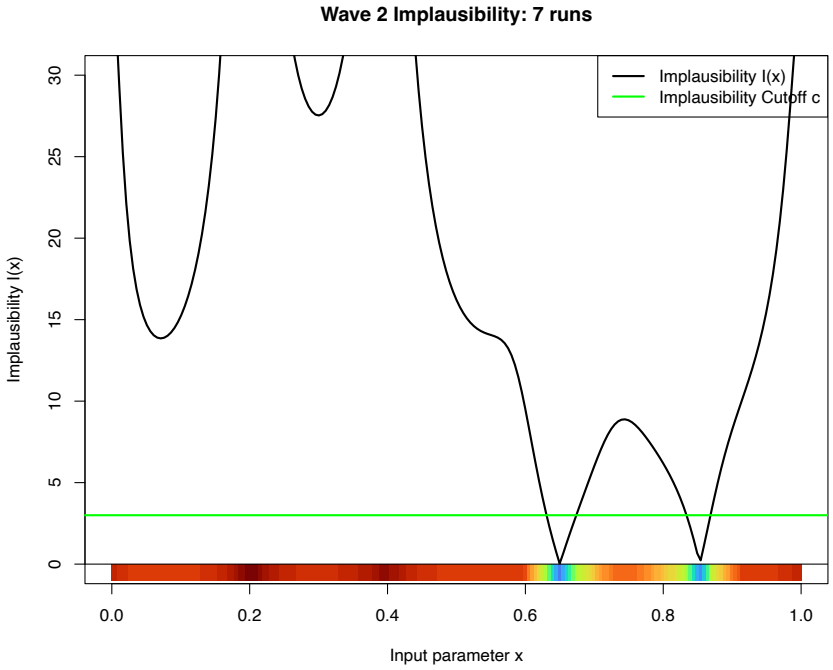


(b) Implausibility measure for use in History Match. Green/blue region of x axis implies $I(x) < 3$.

Figure 19: Wave 1 of a History Matching in 1D: (a) simple emulator, (b) the implausibility measure. Note that inputs for which $I(x) < 3$ are typically viewed as "non-implausible".

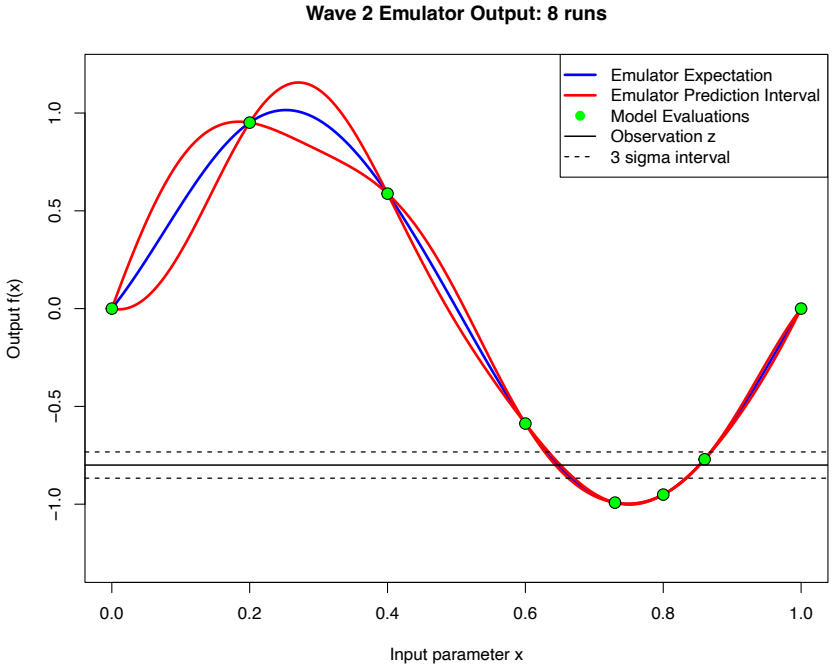


(a) Simple 1D emulator showing observed data $z \pm 3\sqrt{\sigma_e^2 + \sigma_\epsilon^2}$ for use in History Match.

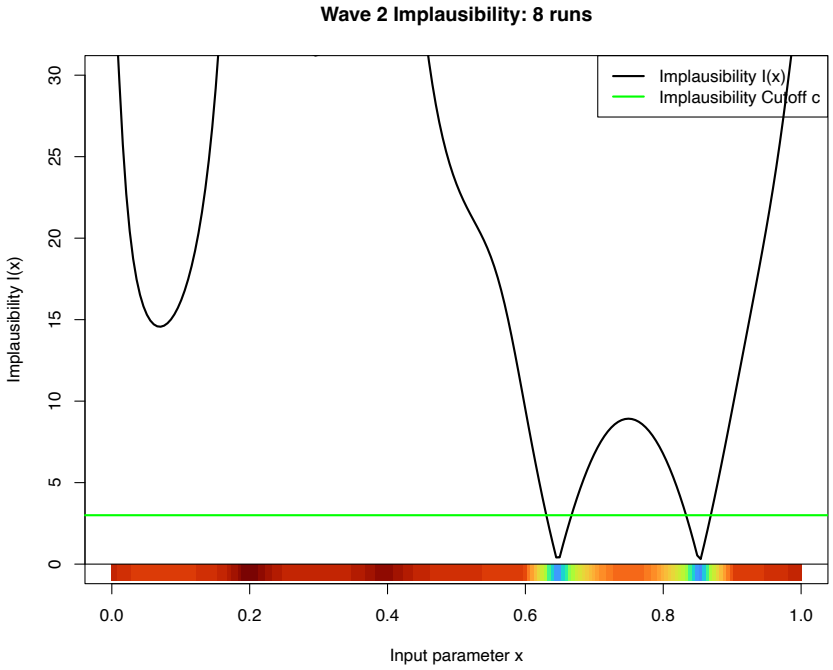


(b) Implausibility measure for use in History Match. Green/blue region of x axis implies $I(x) < 3$.

Figure 20: Wave 2 of a History Matching in 1D: (a) simple emulator, (b) the implausibility measure. Note that inputs for which $I(x) < 3$ are typically viewed as "non-implausible".

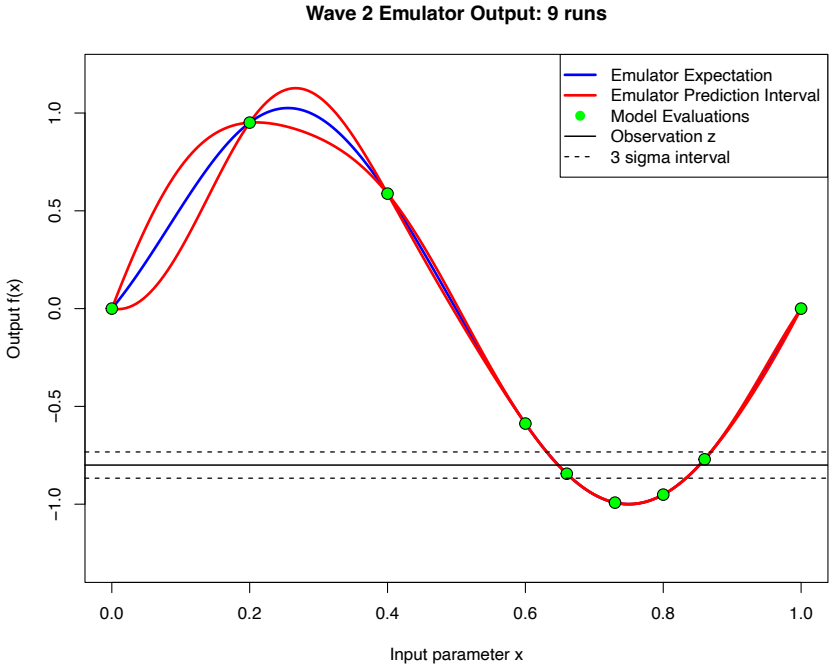


(a) Simple 1D emulator showing observed data $z \pm 3\sqrt{\sigma_e^2 + \sigma_\epsilon^2}$ for use in History Match.

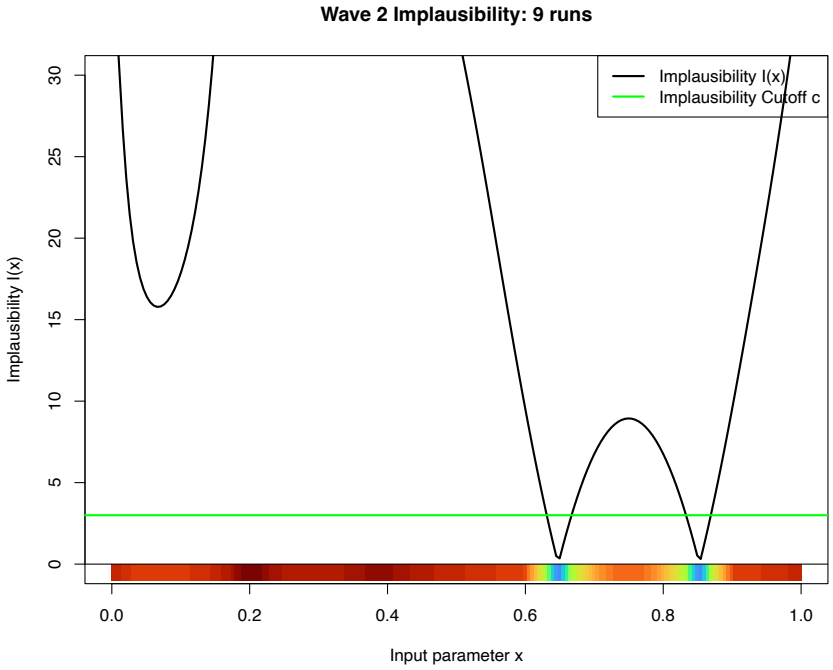


(b) Implausibility measure for use in History Match. Green/blue region of x axis implies $I(x) < 3$.

Figure 21: Wave 2 of a History Matching in 1D: (a) simple emulator, (b) the implausibility measure. Note that inputs for which $I(x) < 3$ are typically viewed as "non-implausible".



(a) Simple 1D emulator showing observed data $z \pm 3\sqrt{\sigma_e^2 + \sigma_\epsilon^2}$ for use in History Match.



(b) Implausibility measure for use in History Match. Green/blue region of x axis implies $I(x) < 3$.

Figure 22: Wave 2 of a History Matching in 1D: (a) simple emulator, (b) the implausibility measure. Note that inputs for which $I(x) < 3$ are typically viewed as "non-implausible".

- When we are prepared to specify the full $q \times q$ covariance structure of the model discrepancy ϵ and the observational errors e we can employ the Multivariate Implausibility which is defined as the following Mahalanobis distance:

$$I_{MV}(x) \quad = \quad (\mathrm{E}_D[f(x)] - z)^T (\mathrm{Var}_D[f(x)] + \mathrm{Var}[\epsilon] + \mathrm{Var}[e])^{-1} (\mathrm{E}_D[f(x)] - z) \quad (73)$$

  where $\mathrm{E}_D[f(x)]$ and z are now length q column vectors and $\mathrm{Var}_D[f(x)]$, $\mathrm{Var}[\epsilon]$ and $\mathrm{Var}[e]$ are $q \times q$ covariance matrices.

- Here we can define $\mathrm{Var}_D[f(x)] = \mathrm{diag}\{\mathrm{Var}_{D_1}[f_1(x)], \ldots, \mathrm{Var}_{D_q}[f_q(x)]\}$ i.e. a diagonal matrix composed of the individual emulator variances or we can use more advanced *multivariate emulators* which we may return to later.

- We may employ such a multivariate measure across all q outputs, or just a subset of r outputs of interest, for example those that are simple to emulate at this stage, or of particular interest.

- Defining a suitable cutoff c_{MV} to use with $I_{MV}(x)$ is harder. A simple heuristic approach can be derived from considering the implausibilities to have a χ^2_r distribution where r is the number of outputs (this follows from an assumption of the normality of $(\mathrm{E}_D[f(x)] - z)$). Thus we could set c_{MV} to be appropriate quantiles of this distribution, or use a more robust distributional assumption e.g. by using the t-distribution.

- Use of the multivariate implausibility measure by imposing the cutoff $I_{MV}(x)$, will often favour runs with the correct shape over the q outputs e.g. for highly correlated MD and OE covariance matrices $I_{MV}(x)$ will favour runs that track the shape of the outputs (runs that are always high or always low) which maybe scientifically desirable compared to runs that get close but have unphysical shape (e.g. over time).

4.4 Iterative History Matching

- History matching of a computer simulator is an inherently iterative process.

- Exploring the full input space to find all the matches between simulator and observed data can be extremely challenging to perform in a single step, therefore a divide-and-conquer strategy is usually of great benefit.

- History Matching uses implausibility measures to cut out regions of implausible input space. Further runs are then performed in the remaining non-implausible space and more space cutout due to increases in emulator accuracy and/or more outputs considered. This is repeated iteratively in a series of *waves*.

- The full HM algorithm is as follows:

Initialisation Phase

1. Define initial input space $\mathcal{X}_0$ e.g by specifying ranges on the input parameters x_i for $i = 1, \ldots, m$.
2. Specify model discrepancy and observational error structures.
3. Set the list of previously emulated outputs $Q_0 = \emptyset$ and set wave number $k = 1$.

Iterative Phase for Wave k

4. Design and evaluate a well chosen set of runs over the current non-implausible space $\mathcal{X}_{k-1}$. e.g. using a maximin Latin hypercube with rejection [1].
5. Scan to see if there are new, informative outputs that can now be emulated accurately (that were difficult to emulate well in previous waves) and add them to the previous set Q_{k-1}, to define Q_k.
6. Construct new, more accurate emulators (typically) defined only over the region $\mathcal{X}_{k-1}$ for each output in Q_k.
7. The implausibility measures $I_i(x)$, $i \in Q_k$, are then recalculated over $\mathcal{X}_{k-1}$ using the new emulators, and combined into total measure $I(x)$ e.g. using $I(x) = I_M(x)$.
8. Cutoffs are imposed on the Implausibility measures e.g. using $I(x) < c_k$, defining a new, smaller non-implausible volume $\mathcal{X}_k$ which should satisfy $\mathcal{X} \subset \mathcal{X}_k \subset \mathcal{X}_{k-1}$.
9. Unless a) the emulator variances for all outputs of interest are now small in comparison to the other sources of uncertainty due to the model discrepancy and observation errors, or b) system properties or forecasts we are interest in will not be impacted by further reduction/resolution of emulator variance, or c) the entire input space has been deemed implausible, or d) computational resources have been exhausted, set $k = k + 1$ and return to step 4.

Final Phase

10. If 9 a) or b) is true, generate a large number of acceptable runs from the final non-implausible volume $\mathcal{X}$, sampled depending on scientific goal.

Example: Iterative History Matching in 1D

- Figure 19b shows the implausibility measure for the 1D example. Note that the x-axis is coloured by implausibility: red/yellow for implausible, green/blue for non-implausible.
- This represents the first wave of the history matching analysis. The reduced non-implausible region $\mathcal{X}_1$ is in green/blue and is defined by

$$\mathcal{X}_1 \equiv \{x \in \mathcal{X}_0 : I(x) < 3\} \tag{74}$$

- We now proceed with wave 2 by performing three more (spaced-out) runs in the non-implausible region only, and updating the emulator and subsequently the implausibility.
- We choose the wave 2 runs to be at: $x_D^{w_2} = (0.73, 0.8, 0.66)$ and the results are shown, with each point included sequentially, in figures 20, 21 and 22.
- Note, the wave 2 runs are spaced-out over $\mathcal{X}_1$ and do not necessarily lie at our best guess for a good match e.g. where $\mathrm{E}_D[f(x)]$ had been equal to z: that would in general be a poor strategy for wave 2 runs (but one that is commonly chosen!).
- We note the following:

  1. The emulator is far more accurate over the regions of interest i.e. where $f(x)$ is close to z, as the prediction interval is far narrower.
  2. The implausibility measure has increased in most places due to the increased accuracy of the emulator, and we are hence able to rule out more input space as implausible.
  3. The current non-implausible region after wave 2:

$$\mathcal{X}_2 \equiv \{x \in \mathcal{X}_1 : I(x) < 3\} \tag{75}$$

     is small and contains good matches to z.
  4. The size of $\mathcal{X}_2$ is now dominated by the OE and MD uncertainties: further runs of the expensive model will not help much as the emulator is already far more accurate than the OE and MD uncertainties. This relates to stopping criteria 9a) of the HM algorithm.
  5. Hence the HM is complete.

Example: Iterative History Matching in 2D

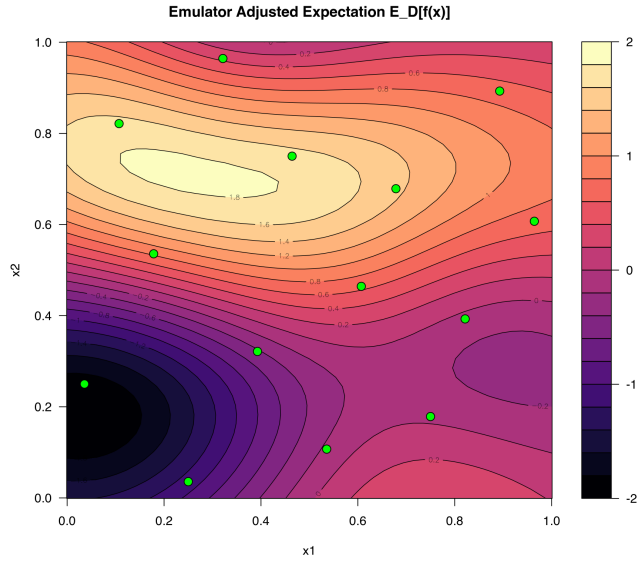
- Consider the 2D emulator of the function defined in section 2.4:

$$f(x) \;=\; f(x_1, x_2) \;=\; -\sin(2\pi x_2) + 0.9\sin(2\pi(1-x_1)(1-x_2)) \qquad (76)$$

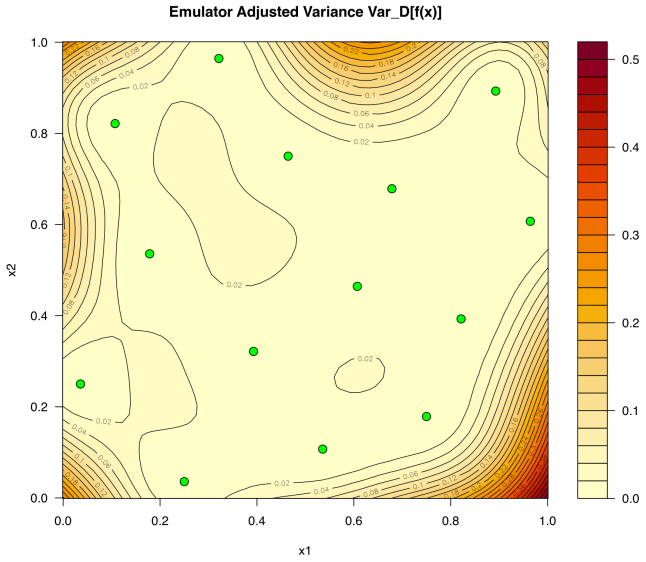
- We perform a wave 1 set of 14 runs using a maxmin LHD and the corresponding emulator expectation and variance are shown in figures 23a and 23b.
- Now imagine we have taken an imperfect observation of the real system y yielding value $z = -1.25$, with OE variance $\mathrm{Var}[e] = 0.06^2$ and MD variance $\mathrm{Var}[\epsilon] = 0.05^2$.
- We can evaluate the implausibility $I(x)$ over a dense 50×50 grid of input points x_P and this is shown in figure 23c.
- The non-implausible region $\mathcal{X}_1$ after wave 1 is seen as the green/blue region inside the $I = 3$ contour line (the thicker black contour line).
- Note that here $\mathcal{X}_1$ is disconnected: along with the obvious part in the bottom left, there are also smaller non-implausible regions in the bottom right and top of the plot.
- We propose 8 new runs for wave 2 shown as the pink candidate points in figure 24. These are chosen to be spread out within $\mathcal{X}_1$.
- These 8 runs are evaluated and then incorporated into the emulator and implausibility in figure 25.
- Note that the non-implausible region $\mathcal{X}_2$ after wave 2 is not disconnected as the two regions in the bottom right and top have been correctly ruled out as implausible. The region in the bottom left is now smaller and defined by the emulator which has low variance over this region.
- As this is a toy model, we can check these results against the implausibility one would obtain were we to replace the emulator expectation $\mathrm{E}_D[f(x)]$ with the true model $f(x)$ with zero emulator uncertainty (i.e. variance). That is we can evaluate the true implausibility due to the model

$$I^2_{true}(x) \;=\; \frac{(f(x) - z)^2}{\mathrm{Var}[e] + \mathrm{Var}[\epsilon]} \qquad (77)$$

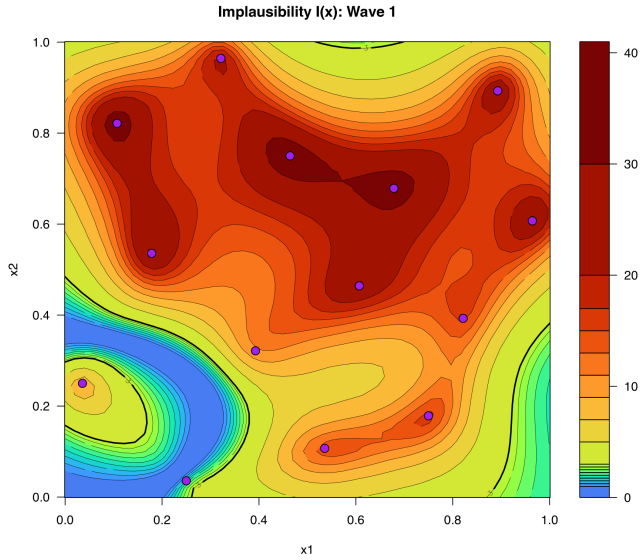
  This is shown in figure 26. We see that the shape and location of $\mathcal{X}_2$ is very similar to the non-implausible region generated by the true function.



(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.

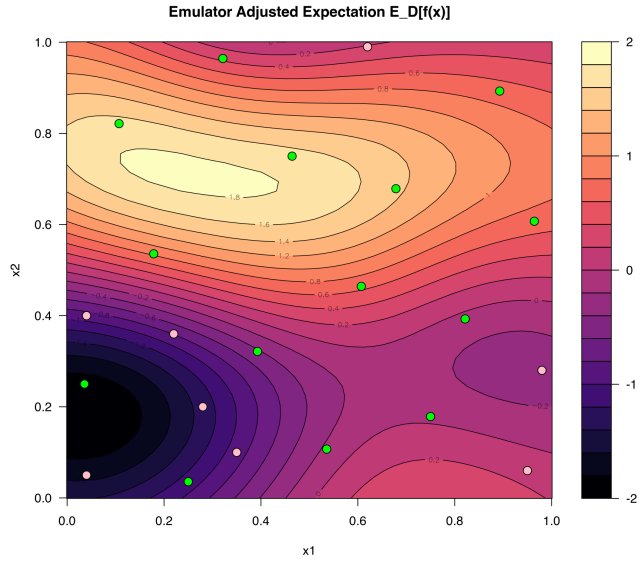


(b) Emulator Variance $\mathrm{Var}_D[f(x)]$.

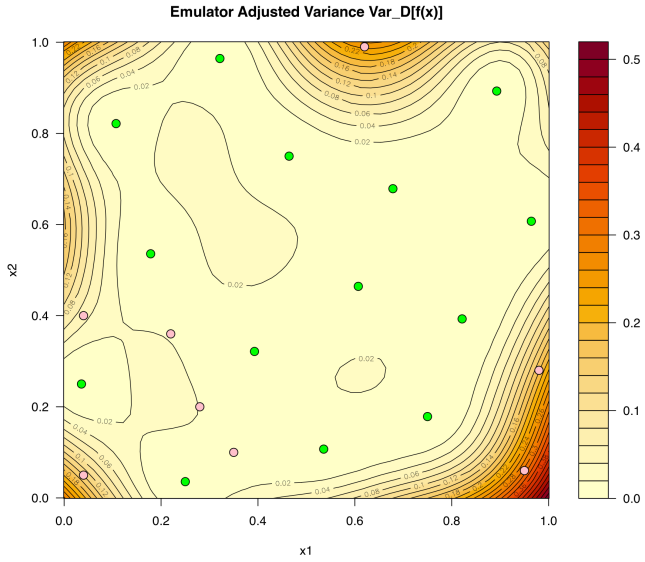


(c) Implausibility $I(x)$, with non-implausible region blue/green.

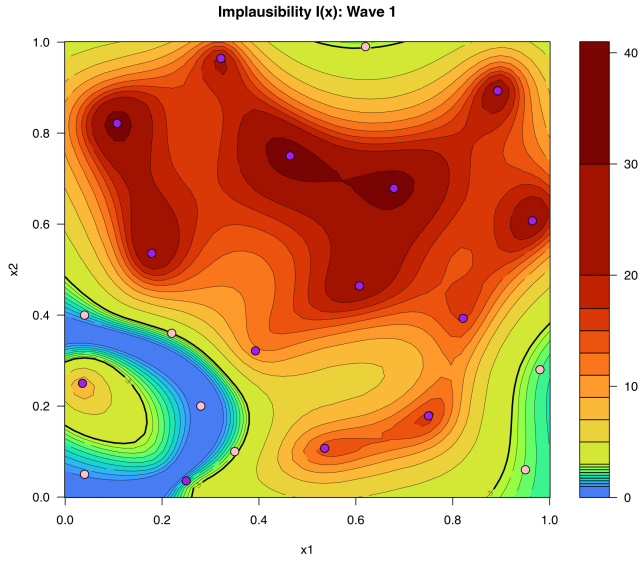
Figure 23: Wave 1 emulator expectation, variance and implausibility, using a maximin LHD of 14 runs (green/purple points).



(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.

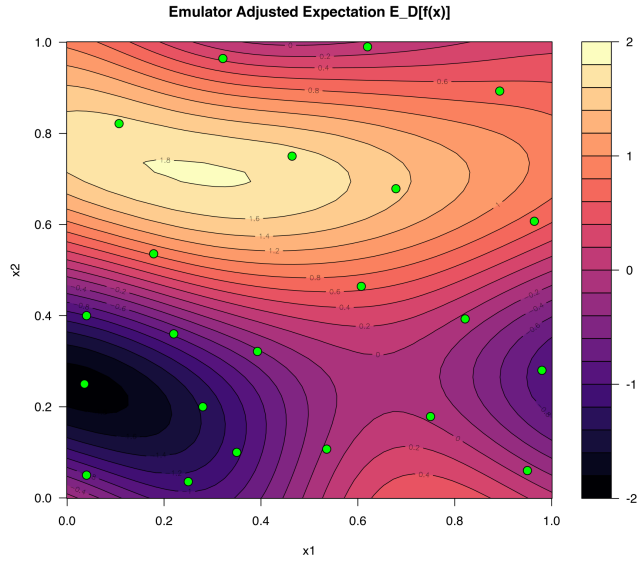


(b) Emulator Variance $\mathrm{Var}_D[f(x)]$.

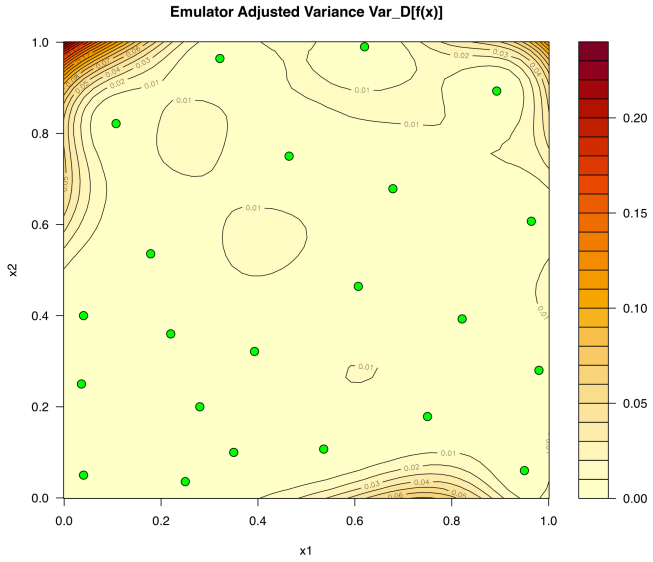


(c) Implausibility $I(x)$ plus candidate wave 2 runs (pink).

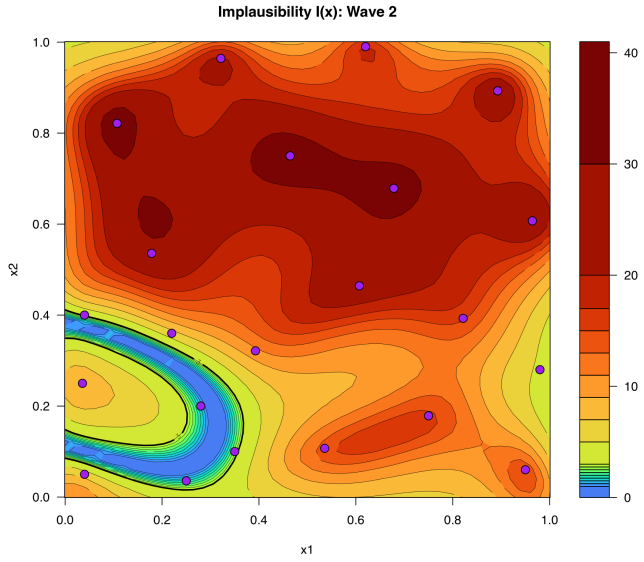
Figure 24: Wave 1 emulator and implausibility using LHD of 14 runs (green/purple points) also showing 8 candidate wave 2 run locations (pink points).



(a) Emulator Expectation $\mathrm{E}_D[f(x)]$.

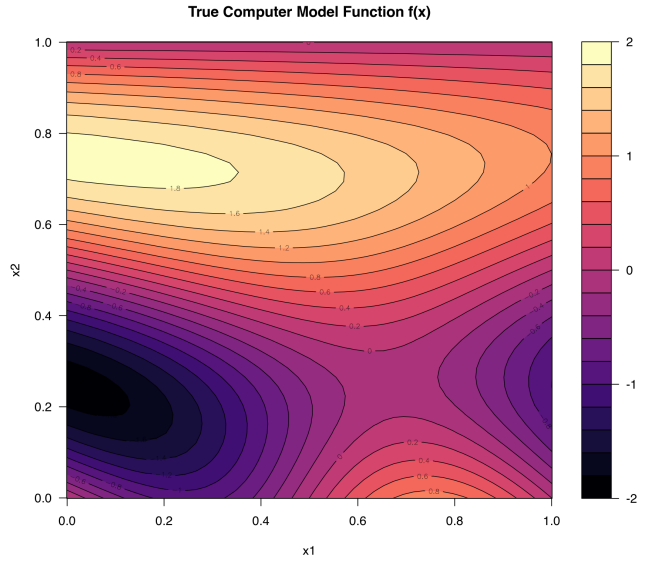


(b) Emulator Variance $\mathrm{Var}_D[f(x)]$.

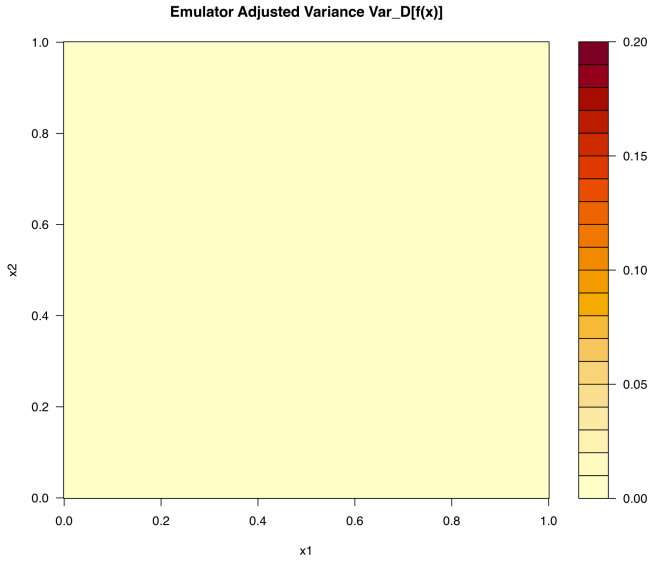


(c) Implausibility $I(x)$ plus candidate wave 2 runs (pink).

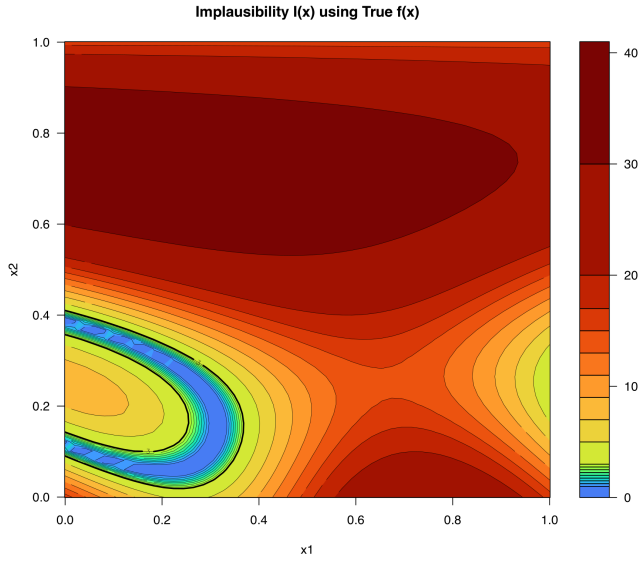
Figure 25: Wave 2 emulator and implausibility using LHD of 14 wave 1 runs and 8 wave 2 runs now evaluated (green/purple points).



(a) The true computer model function $f(x)$.



(b) The variance of the true model, i.e. zero.



(c) Implausibility $I(x)$ using the true $f(x)$.

Figure 26: (a) The true function $f(x)$ for comparison with the previous two figures. (b) its corresponding uncertainty which is zero, and (c) the implausibility using the true $f(x)$.

Why History Matching Works (for simple emulators)

- HM has many strengths, some of which exploit features of more complex emulators that we are yet to introduce.
- For now, we can state the following:

  1. HM is efficient as it employs (batch) sequential design (aka active learning) where we see the results of the first wave, construct an emulator, and use it to decide where best to put the next wave 2 batch of runs.
  2. The design for subsequent waves is chosen based on both the emulator expectation (our best guess) and emulator variance (how uncertain we are).
  3. As we have a concept of the various uncertainties incorporated into the implausibility measure we can rule out uninteresting parts of the input space on a point by point basis (in contrast to a standard Bayesian posterior).
  4. As we zoom into smaller regions, it becomes easier to emulate as a deterministic function typically becomes smoother (lots more to say about this!).

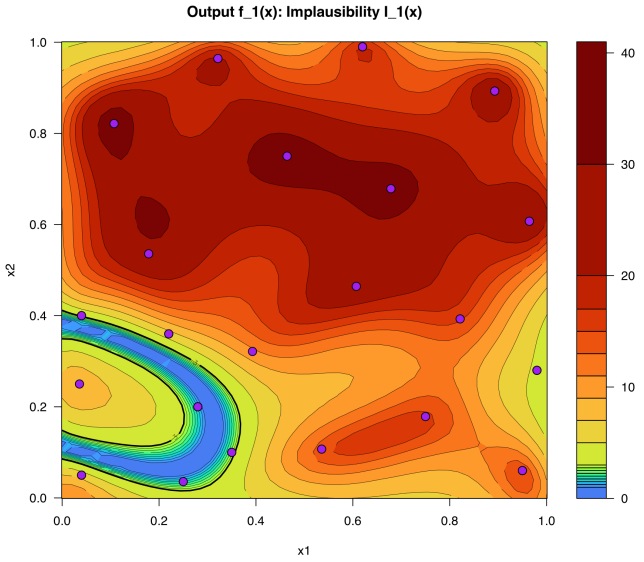
Example: Iterative History Matching with Multiple Outputs

- The most powerful use of History Matching is when there are multiple outputs $f_i(x), i = 1, \ldots, q$.
- Say we have a computer model with two outputs and two inputs so $f \in \mathbb{R}^2$, $x \in \mathbb{R}^2$:

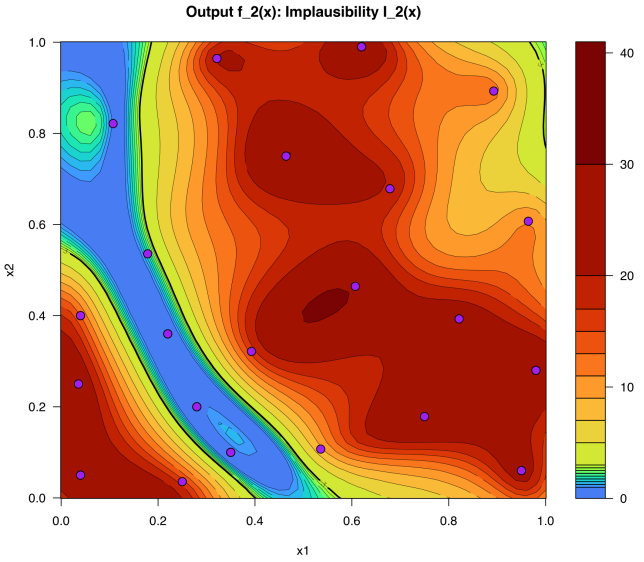
$$f_1(x) = -\sin(2\pi x_2) + 0.9\sin(2\pi(1-x_1)(1-x_2)) \quad (78)$$
$$f_2(x) = 2\sin(2\pi(x_2/2 + (x_1-1)^2 + (x_2-1)^2)) \quad (79)$$

  and we have measured observed data $z = (z_1, z_2)^T = (-1.25, 1.9)^T$ with $\mathrm{Var}[e_1] = 0.06^2$, $\mathrm{Var}[e_2] = 0.1^2$, $\mathrm{Var}[\epsilon_1] = 0.05^2$, $\mathrm{Var}[\epsilon_2] = 0.08^2$, and define the input space to be $\mathcal{X}_0 = [0,1]^2$.

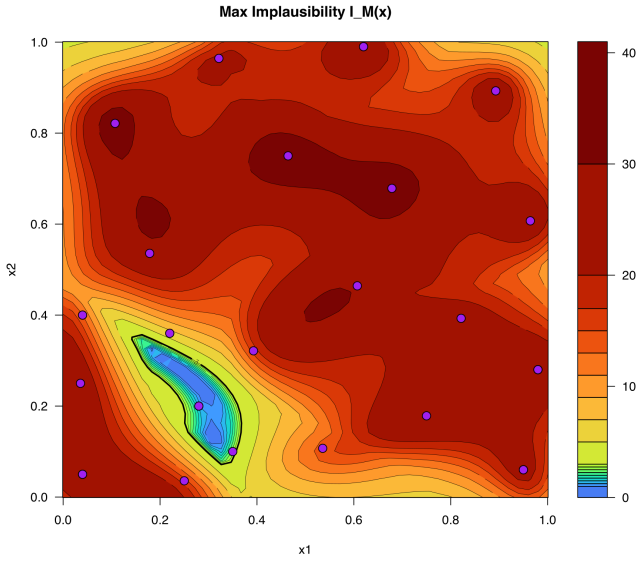
- The implausibilities $I_1(x)$ (shown previously) and $I_2(x)$ (new) are shown in figures 27a and 27b respectively, and show the constraints placed on the input space $\mathcal{X}_0$ by the two observations z_1 and z_2.
- We can easily construct the maximised implausibility $I_M(x) = \max_i I_i(x)$, which is given in figure 27c, which shows a much reduced non-implausible region.
- Note that even if the emulator for $f_2(x)$ was not that accurate in certain regions of the input space (e.g. the top left) this does not matter if those regions are already ruled out by the implausibility corresponding to $f_1(x)$: we *don't* need to emulate *every output* accurately over *all of the input space* $\mathcal{X}$: a key History Matching principle.



(a) Implausibility $I_1(x)$ for output $f_1(x)$.



(b) Implausibility $I_2(x)$ for output $f_2(x)$.



(c) Max Implausibility $I_M(x)$

Figure 27: The combination of implausibilities from two outputs $f_1(x)$ and $f_2(x)$: (a) $I_1(x)$, (b) $I_2(x)$ and (c) the maximum implausibility $I_M(x) = \max_i I_i(x)$.

4.5 Visualising the History Match in Higher Dimensions

Minimised Implausibility Projections $I_P(x')$

- When the dimension m of $\mathcal{X}$ is 3 or more then visualising the non-implausible region becomes a more challenging problem.
- One approach is to project the implausibility function from the high-dimensional space into simpler 2-dimensional sub-spaces corresponding to every pair of inputs.
- Suppose $x' = (x'_1, x'_2)$ is the pair of inputs onto which we wish to project the implausibility, then we partition the m-dimensional inputs x into (x', x'') where x'' is the $(m-2)$-dimensional remainder.
- The implausibility function $I(x)$ can then be collapsed from the entirety of $\mathcal{X}$ onto the subspace of x' by calculating:

$$I_P(x') = \min_{x''} I\left[(x', x'')\right], \tag{80}$$

  which is a function of x' only.
- Under this *minimised implausibility* projection, a value of the input pair x' is judged to be implausible if and only if it is implausible in combination with *all* possible values of the $(m-2)$-dimensional x''.
- Conversely, if any combination of x' and x'' is non-implausible then x' is also non-implausible.

Optical Depth Projections $\rho(x')$

- The above minimised implausibility projection is very useful, but essentially projects the silhouette of the non-implausible region $\mathcal{X}_k$ for various cutoffs and gives no information about the depth or thickness of the non-implausible region $\mathcal{X}_k$ in the remaining $(m-2)$-dimensional subspace spanned by x''.
- Instead, we quantify this *optical depth*, $\rho(x')$, of the current non-implausible region $\mathcal{X}_k$ in the x'' space conditioned on fixed x', as simply the proportion of the $(m-2)$-dimensional x'' space deemed non-implausible for fixed input subset x'.
- At wave k we compute $\rho(x')$ as

$$\rho(x') = \frac{V_{m-2}\{x \in \mathcal{X}_k \,|\, x'\}}{V_{m-2}\{x \in \mathcal{X} \,|\, x'\}}, \tag{81}$$

  which is 0 when there are no non-implausible points possible (with any choice of x'') for fixed x', and is 1 when we find exclusively non-implausible points for fixed x'

(whatever the choice of x'') and where we define $\mathrm{V}_{m-2}\{.\}$ as the $(m-2)$-dimensional volume of the input space, here conditioned on a fixed value of the 2-dimensional x'.

- This optical depth can then show where large or small amounts of non-implausible points can be found along that direction defined by fixed x', providing further insight into the structure of $\mathcal{X}_k$.
- Direct analytic evaluation of $I_P(x')$ and $\rho(x')$ is usually impossible, but we can accurately approximate them using carefully chosen *emulator evaluation designs* for x_P e.g. grid designs in lower dimensions or large LHD in higher dimensions.

References

[1] Vernon, I., Goldstein, M., Bower, R.G.: Galaxy formation: a bayesian uncertainty analysis. Bayesian Analysis **5**(4), 619–670 (2010)

[2] Vernon, I., Goldstein, M., Bower, R.G.: Rejoinder for Galaxy formation: a bayesian uncertainty analysis. Bayesian Analysis **5**(4), 697–708 (2010)

[3] Bower, R.G., Vernon, I., Goldstein, M., Benson, A.J., Lacey, C.G., Baugh, C.M., Cole, S., Frenk, C.S.: The parameter space of galaxy formation. Mon.Not.Roy.Astron.Soc. **96**(454), 717–729 (2010)

[4] Vernon, I., Goldstein, M., Bower, R.G.: Galaxy formation: Bayesian history matching for the observable universe. Statistical Science **29**(1), 81–90 (2014)

[5] Rodrigues, L.F.S., Vernon, I., Bower, R.G.: Constraints to galaxy formation models using the galaxy stellar mass function. MNRAS **466**(2), 2418–2435 (2017)

[6] Scarponi, D., Iskauskas, A., Clark, R.A., Vernon, I., McKinley, T.J., Goldstein, M., Mukandavire, C., Deol, A., Weerasuriya, C., Bakker, R., White, R.G., McCreesh, N.: Demonstrating multi-country calibration of a tuberculosis model using new history matching and emulation package - hmer. Epidemics **43**, 100678 (2023). doi:10.1016/j.epidem.2023.100678

[7] Vernon, I., Owen, J., Aylett-Bullock, J., Cuesta-Lazaro, C., Frawley, J., A., A.Q.-B., Sedgewick, Shi, D., Truong, H., Turner, M., Walker, J., Caulfield, T., Fong, K., Krauss, F.: Bayesian emulation and history matching of june. Phil. Trans. R. Soc. A **380**(2233) (2022)

[8] Andrianakis, I., Vernon, I., McCreesh, N., McKinley, T.J., Oakley, J.E., Nsubuga, R., Goldstein, M., White, R.G.: Bayesian history matching of complex infectious disease models using emulation: A tutorial and a case study on HIV in uganda. PLoS Comput Biol. **11**(1), 1003968 (2015)

[9] Andrianakis, I., Vernon, I., McCreesh, N., McKinley, T.J., Oakley, J.E., Nsubuga, R.N., Goldstein, M., White, R.G.: History matching of a complex epidemiological model of human immunodeficiency virus transmission by using variance emulation. Journal of the Royal Statistical Society: Series C (Applied Statistics) **66**(4), 717–740 (2017). doi:10.1111/rssc.12198

[10] Andrianakis, I., McCreesh, N., Vernon, I., McKinley, T.J., Oakley, J.E., Nsubuga, R., Goldstein, M., White, R.G.: Efficient history matching of a high dimensional individual based hiv transmission model. SIAM/ASA Journal of Uncertainty Quantification **5**(1), 694–719 (2017)

[11] McCreesh, N., Andrianakis, I., Nsubuga, R.N., Strong, M., Vernon, I., McKinley, T.J., Oakley, J.E., Goldstein, M., Hayes, R., White, R.G.: Universal test, treat, and keep: improving art retention is key in cost-effective hiv control in uganda. BMC Infectious Diseases **17**(1), 322 (2017). doi:10.1186/s12879-017-2420-y

[12] Williamson, D., Goldstein, M., Allison, L., Blaker, A., Challenor, P., Jackson, L., Yamazaki, K.: History matching for exploring and reducing climate model parameter space using observations and a large perturbed physics ensemble. Climate Dynamics **41**(7-8), 1703–1729 (2013)

[13] James M. Salter, J.S. Daniel B. Williamson, Kharin, V.: Uncertainty quantification for computer models with spatial output using calibration-optimal bases. Journal of the American Statistical Association **114**(528), 1800–1814 (2019). doi:10.1080/01621459.2018.1514306. https://doi.org/10.1080/01621459.2018.1514306

[14] Hourdin, F., Ferster, B., Deshayes, J., Mignot, J., Musat, I., Williamson, D.: Toward machine-assisted tuning avoiding the underestimation of uncertainty in climate change projections. Science Advances **9**(29), 2758 (2023). doi:10.1126/sciadv.adf2758. https://www.science.org/doi/pdf/10.1126/sciadv.adf2758

[15] Baker, E., Harper, A.B., Williamson, D., Challenor, P.: Emulation of high-resolution land surface models using sparse gaussian processes with application to jules. Geoscientific Model Development **15**(5), 1913–1929 (2022). doi:10.5194/gmd-15-1913-2022

[16] Craig, P.S., Goldstein, M., Seheult, A.H., Smith, J.A.: Bayes linear strategies for history matching of hydrocarbon reservoirs. In: Bernardo, J.M., Berger, J.O., Dawid, A.P., Smith, A.F.M. (eds.) Bayesian Statistics 5, pp. 69–95. Clarendon Press, Oxford, UK (1996)

[17] Craig, P.S., Goldstein, M., Seheult, A.H., Smith, J.A.: Pressure matching for hydrocarbon reservoirs: a case study in the use of bayes linear strategies for large computer experiments (with discussion). In: Gatsonis, C., Hodges, J.S., Kass, R.E., McCulloch,

R., Rossi, P., Singpurwalla, N.D. (eds.) Case Studies in Bayesian Statistics vol. 3, pp. 36–93. Springer, New York (1997)

[18] Cumming, J.A., Goldstein, M.: Bayes linear uncertainty analysis for oil reservoirs based on multiscale computer experiments. In: O'Hagan, A., West, M. (eds.) The Oxford Handbook of Applied Bayesian Analysis, pp. 241–270. Oxford University Press, Oxford, UK (2010)

[19] Cumming, J.A., Goldstein, M.: Small sample bayesian designs for complex high-dimensional models based on information gained using fast approximations. Technometrics **51**(4), 377–388 (2009)

[20] Jackson, S.E., Vernon, I., Liu, J., Lindsey, K.: Understanding hormonal crosstalk in arabidopsis root development via emulation and history matching. Statistical Applications in Genetics and Molecular Biology **19**(2), 20180053 (2020). doi:10.1515/sagmb-2018-0053

[21] Vernon, I., Liu, J., Goldstein, M., Rowe, J., Topping, J., Lindsey, K.: Bayesian uncertainty analysis for complex systems biology models: emulation, global parameter searches and evaluation of gene functions. BMC Systems Biology **12**(1), 1607–06358 (2018)

[22] Goldstein, M., Seheult, A., Vernon, I.: Assessing Model Adequacy. In: Wainwright, J., Mulligan, M. (eds.) Environmental Modelling: Finding Simplicity in Complexity, 2nd edn. John Wiley & Sons, Ltd, Chichester, UK (2013). doi:10.1002/9781118351475.ch26

[23] Goldstein, M., Huntley, N.: In: Ghanem, R., Higdon, D., Owhadi, H. (eds.) Bayes Linear Emulation, History Matching, and Forecasting for Complex Computer Simulators, pp. 1–24. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-11259-6_14-1

[24] Boukouvalas, A., Sykes, P., Cornford, D., , Maruri-Aguilar, H.: Bayesian precalibration of a large stochastic microsimulation model. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS **15**(3) (2014)

[25] Hu, B., Jiang, W., Miyagi, T., Sun, Z., Ekström, A., Forssén, C., Hagen, G., Holt, J.D., Papenbrock, T., Stroberg, S.R., Vernon, I.: Ab initio predictions link the neutron skin of 208pb to nuclear forces. Nature Physics **18**(10), 1196–1200 (2022). doi:10.1038/s41567-022-01715-8

[26] Kondo, Y., Achouri, N.L., et. al.: First observation of 28o. Nature **620**(7976), 965–970 (2023). doi:10.1038/s41586-023-06352-6

[27] Edwards, T.L., Brandon, M.A., Durand, G., Edwards, N.R., Golledge, N.R., Holden, P.B., Nias, I.J., Payne, A.J., Ritz, C., Wernecke, A.: Revisiting antarctic ice loss due to marine ice-cliff instability. Nature **566**(7742), 58–64 (2019). doi:10.1038/s41586-019-0901-4

[28] Pukelsheim, F.: The three sigma rule. The American Statistician **48**, 88–91 (1994)

[29] Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge, Massachusetts (2006)

[30] Kennedy, M.C., O'Hagan, A.: Bayesian calibration of computer models. Journal of the Royal Statistical Society, Series B **63**(3), 425–464 (2001)

5 Advanced Emulators: Dealing with Higher Dimensional Inputs.

5.1 Problems with Higher Dimensional Inputs

- The simple emulator we have explored (based on simple covariance structures) is great for lower input dimension scenarios, however it starts to struggle in higher dimensions (e.g. $m > 10$ say).
- This is because the predictive capability of the emulator at a new input point x depends primarily on the proximity of x to the nearest few runs. If there are no runs close by, the predictions for $f(x)$ will resort to the prior for $f(x)$ as we have seen.
- The problem is that high-dimensional input spaces are large! So often x may not be close to any runs at all. In other words, there may be large holes in our design e.g. in $m = 20$ there are over 10^6 corners, which if evaluated, would still leave a massive hole in the middle of the input space!
- This issue can be only partially mitigated by careful assessment of the correlation lengths θ and having different correlation lengths θ_i in different directions but more on this later.
- Before we develop more advanced emulator structures, it is worth considering the difference between Bayes Linear emulation and standard linear regression.

5.2 Reminder: Linear Regression

- In standard linear regression we choose a set of p basis functions e.g. $\{1, x_1, x_2, x_1^2, x_2^2, x_1x_2\}$ for $x \in \mathbb{R}^2$ and try to mimic some observations of a real world process using a linear combination of these basis functions, with unknown coefficients β_j, $j = 1, \ldots, p$.
- We also assume the real world process has been observed with error e which gives rise to noisy observations here (temporarily) denoted $f(x)$.
- Representing the vector of basis functions as $g(x) = \{1, x_1, x_2, x_1^2, x_2^2, x_1x_2\}$ so e.g. $g_2(x) = x_1$ and $g_6(x) = x_1x_2$ we can write the statistical model as

$$f(x) \;=\; \sum_{j=1}^{p} \beta_j g_j(x) \;+\; e \tag{82}$$

where e is typically assumed to have mean zero and variance σ_e^2 and to be independent for different values of x so $\mathrm{Cov}[e(x), e(x')] = 0$ for $x \neq x'$.

- Linear regression is very effective and widely used (a candidate for the most used equation of all time!).
- So why didn't we just use regression for our emulators? Well, regression has many desirable features but its weaknesses include:
  1. Independent error structure: hence has quite a simple variance structure.
  2. Its predictions cannot interpolate points perfectly and the variance does not shrink to zero close to a known run, unlike our BL emulators: i.e. it is not set up to deal with deterministic (i.e. repeatable) model output.
  3. Not so flexible: it cannot deal with arbitrary shapes in the output. We need to choose the basis functions carefully.
- However, we would really like to incorporate some of regressions desirable features:
  1. It is extremely fast to evaluate (requires inverting a $p \times p$ matrix).
  2. It fits global basis functions and so can capture global structure using very few runs: can deal with larger number of input dimensions.
  3. Exhibits modest extrapolation capabilities (in some cases).
  4. Components easy to interpret scientifically.
  5. Lots of standard diagnostics to warn of any problems.
- Reminder: classical i.e. frequentist estimates $\hat{\beta}$ for the vector of $\beta = (\beta_1, \ldots, \beta_p)^T$ coefficients are found from Ordinary Least Squares (OLS) and are given by:

$$\hat{\beta}_{OLS} = (X^T X)^{-1} X^T D \qquad \text{with} \qquad \text{Var}[\hat{\beta}_{OLS}] = \sigma_e^2 (X^T X)^{-1} \tag{83}$$

  where X is the $n \times p$ *design matrix* with rows formed by the vector of basis functions $g(x^{(j)})^T$ at each of the n input locations $x^{(j)}$ and D is the vector of n observations:

$$X = \begin{pmatrix} g(x^{(1)})^T \\ \vdots \\ g(x^{(n)})^T \end{pmatrix} = \begin{pmatrix} g_1(x^{(1)}) & \cdots & g_p(x^{(1)}) \\ \vdots & \vdots & \vdots \\ g_1(x^{(n)}) & \cdots & g_p(x^{(n)}) \end{pmatrix}, \qquad D = \begin{pmatrix} f(x^{(1)}) \\ \vdots \\ f(x^{(n)}) \end{pmatrix} \tag{84}$$

- A generalisation to correlated observational error e exists and is known as Generalised Least Squares (GLS) where $\text{Cov}[e(x^{(i)}), e(x^{(j)})] = \Omega_{ij}$ for some $n \times n$ covariance matrix Ω. In this case the estimates change to (the Ω^{-1} essentially decorrelates the errors):

$$\hat{\beta}_{GLS} = (X^T \Omega^{-1} X)^{-1} X^T \Omega^{-1} D \qquad \text{with} \qquad \text{Var}[\hat{\beta}_{GLS}] = (X^T \Omega^{-1} X)^{-1} \tag{85}$$

- It will be insightful to see if there is a connection between our more advanced emulators and the above OLS and GLS results.

5.3 Advanced Emulator Part 1: Including Regression Terms

- Helpfully, we can incorporate a regression structure into our Bayes Linear emulator.
- It will be included in a Bayesian way i.e. also using prior statements regarding the regression coefficients β_j, $j = 1, \ldots, p$ (similar to full Bayesian regression, but unlike in classical regression which just estimates β_j using OLS and no priors).
- Instead of our standard weakly stationary process emulator $f(x) = u(x)$ we now also include regression terms with basis functions $g_j(x)$ and unknown coefficients β_j:

$$f(x) \;=\; \sum_{j=1}^{p} \beta_j g_j(x) \;+\; u(x) \tag{86}$$

  (where $f(x)$ is back to representing our usual output from a deterministic computer model).

- As they are unknown, we need to specify prior expectations $\mathrm{E}[\beta_j]$ and covariances $\mathrm{Cov}[\beta_j, \beta_k]$ for the β_j coefficients. Denoting the column vector of coefficients as $\beta = (\beta_1, \ldots, \beta_p)^T$, these prior statements can be grouped into the length p column vector:

$$\mathrm{E}[\beta] \;=\; \mu_\beta \tag{87}$$

  and the $p \times p$ covariance matrix:

$$\mathrm{Var}[\beta] \;=\; \Sigma_\beta \tag{88}$$

  which will be used shortly.

- We need to choose scientifically appropriate basis functions, but often a general choice is low order polynomials e.g. full quadratics in 2D would give $g(x) = \{1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\}$ etc. We also (typically) judge the β_j to be uncorrelated with $u(x)$, a priori.
- We still give the weakly stationary process $u(x)$ one of the standard stationary covariance structures discussion in section 2.6 as before e.g.

$$\mathrm{Cov}[u(x), u(x')] \;=\; \sigma^2 c(x - x') \;=\; \sigma^2 \exp\left\{ -\frac{||x - x'||^2}{\theta^2} \right\} \tag{89}$$

  which will ensure the property of interpolating known runs, with the emulator variance shrinking to zero close to them, as we shall see.

- Note however that now the covariance structure of $f(x)$ itself is altered compared to the simple emulator (see equation (5)). Denoting $g(x) = (g_1(x), \ldots, g_p(x))^T$ we have:

$$\begin{aligned} f(x) &= \sum_{j=1}^{p} \beta_j g_j(x) \; + \; u(x) && (90)\\ &= g(x)^T \beta \; + \; u(x) && (91)\\ \Rightarrow \qquad \mathrm{Cov}[f(x), f(x')] &= \mathrm{Cov}[g(x)^T \beta \; + \; u(x), g(x')^T \beta \; + \; u(x')] && (92)\\ &= g(x)^T \mathrm{Cov}[\beta, \beta] g(x') \; + \; \mathrm{Cov}[u(x), u(x')] && (93)\\ &= g(x)^T \mathrm{Var}[\beta] g(x') \; + \; \sigma^2 c(x - x') && (94)\\ &= g(x)^T \Sigma_\beta \, g(x') \; + \; \sigma^2 c(x - x') && (95) \end{aligned}$$

  (using $\mathrm{Cov}[\beta, u(x)] = 0$) so $f(x)$ is no longer stationary, and has a lot more global structure induced via the basis functions $g(x)$.

- The prior expectation of $f(x)$ has also changed and now given by (again assuming $\mathrm{E}[u(x)] = 0$):

$$\begin{aligned} \mathrm{E}[f(x)] &= \mathrm{E}[g(x)^T \beta \; + \; u(x)] && (96)\\ &= g(x)^T \mathrm{E}[\beta] + 0 && (97)\\ &= g(x)^T \mu_\beta && (98) \end{aligned}$$

  so now we can incorporate prior knowledge about the physical structure of $f(x)$ from the scientific context e.g. if we knew that $f(x)$ will probably increase with increasing input x_2 say, we would choose $\mathrm{E}[\beta_3]$ (the coefficient of x_2) to be positive, and $\mathrm{Var}[\beta_3]$ to be relatively small (depending how sure we were).

- This ability to build in prior knowledge via the regression terms is important in many settings (but not always), especially when performing *multilevel emulation*, where we build emulators for simulators of differing levels of accuracy.

5.4 Advanced Emulator Part 1: Applying the Bayes Linear Update

- We now perform a wave 1 set of n runs of the simulator at locations $x_D = (x^{(1)}, \ldots, x^{(n)})$ to obtain $D = (f(x^{(1)}), f(x^{(2)}), \ldots, f(x^{(n)}))^T$ as before.

- As we have the expectation and covariance structure of $f(x)$, we can just apply the Bayes linear update given by equations (9) and (10) to obtain $\mathrm{E}_D[f(x)]$ and $\mathrm{Var}_D[f(x)]$ at new x as usual.

- However, by the linearity of the BL update we will of course have that:

$$\mathrm{E}_D[f(x)] \;=\; \mathrm{E}_D[g(x)^T \beta \; + \; u(x)] \;=\; g(x)^T \mathrm{E}_D[\beta] \; + \; \mathrm{E}_D[u(x)] \qquad (99)$$

and it will be deeply insightful to examine the update to the regression coefficients β and the weakly stationary process $u(x)$ seperately. Before we do this, we need to prove some matrix identities.

Extremely Useful Matrix Identities part 1

Theorem: Matrix Identity 1 (**MI1**)

- For matrices P and Q and assuming the inverses exist

$$P(QP+I)^{-1} = (PQ+I)^{-1}P \tag{100}$$

Proof: Matrix Identity 1 (**MI1**)

- For matrices P and Q and assuming the following inverses exist:

$$\begin{aligned} PQP+P &= (PQ+I)P = P(QP+I) \\ \Rightarrow \quad P(QP+I)^{-1} &= (PQ+I)^{-1}P \end{aligned}$$

where to get the last line we have just multiplied the middle and right expression from the first line by $(QP+I)^{-1}$ from the right and by $(PQ+I)^{-1}$ from the left.

Theorem: Matrix Identity 2 (**MI2**)

- For invertible matrices A and C of dimensions $p \times p$ and $n \times n$ respectively, and matrices B and D of appropriate dimension (i.e. $p \times n$ and $n \times p$ respectively):

$$AB\,(DAB+C)^{-1} = (BC^{-1}D+A^{-1})^{-1}BC^{-1} \tag{101}$$

Proof: Matrix Identity 2 (**MI2**)

- We have

$$\begin{aligned} AB\,(DAB+C)^{-1} &= AB\,[C(C^{-1}DAB+I)]^{-1} \\ &= AB\,(C^{-1}DAB+I)^{-1}C^{-1} \\ &= (ABC^{-1}D+I)^{-1}ABC^{-1} \qquad \text{(using MI1)} \\ &= [A(BC^{-1}D+A^{-1})]^{-1}ABC^{-1} \\ &= (BC^{-1}D+A^{-1})^{-1}A^{-1}ABC^{-1} \\ &= (BC^{-1}D+A^{-1})^{-1}BC^{-1} \end{aligned}$$

- This is an **extremely useful matrix identity** and we will use it a lot! See also the "Sherman-Morrison-Woodbury Identity", another very useful matrix inverse result, which we will prove later.

Adjusted Expectation and Variance of β: $\mathrm{E}_D[\beta]$ and $\mathrm{Var}_D[\beta]$

- Before we embark on some large calculations, lets summarise some notation. We have as defined previously:

$$D = \begin{pmatrix} f(x^{(1)}) \\ \vdots \\ f(x^{(n)}) \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \mathrm{E}[\beta] = \mu_\beta, \quad \mathrm{Var}[\beta] = \Sigma_\beta \tag{102}$$

but we also need the vector U composed of $u(x)$ at each of the n runs:

$$U = \begin{pmatrix} u(x^{(1)}) \\ \vdots \\ u(x^{(n)}) \end{pmatrix} \quad \Rightarrow \quad \mathrm{E}[U] = \begin{pmatrix} \mathrm{E}[u(x^{(1)})] \\ \vdots \\ \mathrm{E}[u(x^{(n)})] \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{103}$$

$$\begin{aligned} \Rightarrow \quad \mathrm{Var}[U] &= \begin{pmatrix} \mathrm{Var}[u(x^{(1)})] & \mathrm{Cov}[u(x^{(1)}), u(x^{(2)})] & \cdots & \mathrm{Cov}[u(x^{(1)}), u(x^{(n)})] \\ \mathrm{Cov}[u(x^{(2)}), u(x^{(1)})] & \mathrm{Var}[u(x^{(2)})] & \cdots & \mathrm{Cov}[u(x^{(2)}), u(x^{(n)})] \\ \vdots & \vdots & \vdots & \vdots \\ \mathrm{Cov}[u(x^{(n)}), u(x^{(1)})] & \mathrm{Cov}[u(x^{(n)}), u(x^{(2)})] & \cdots & \mathrm{Var}[u(x^{(n)})] \end{pmatrix} \\ &= \sigma^2 \begin{pmatrix} c(x^{(1)}, x^{(1)}) & c(x^{(1)}, x^{(2)}) & \cdots & c(x^{(1)}, x^{(n)}) \\ c(x^{(2)}, x^{(1)}) & c(x^{(2)}, x^{(2)}) & \cdots & c(x^{(2)}, x^{(n)}) \\ \vdots & \vdots & \vdots & \vdots \\ c(x^{(n)}, x^{(1)}) & c(x^{(n)}, x^{(2)}) & \cdots & c(x^{(n)}, x^{(n)}) \end{pmatrix} \qquad (104) \\ &\equiv \Omega \qquad (105) \end{aligned}$$

defining Ω to be the $n \times n$ covariance matrix of U composed of all the pair wise kernel evaluations $\sigma^2 c(x^{(i)}, x^{(j)})$ of the run locations. Note that this is similar to the matrix we used for $\mathrm{Var}[D]$ in the simple emulator, but now $\mathrm{Var}[D]$ will be more complex.

- We also again need the $n \times p$ "design matrix" X with rows formed by the vector of basis functions $g(x^{(j)})^T$ at each run $x^{(j)}$:

$$X = \begin{pmatrix} g(x^{(1)})^T \\ \vdots \\ g(x^{(n)})^T \end{pmatrix} = \begin{pmatrix} g_1(x^{(1)}) & \cdots & g_p(x^{(1)}) \\ \vdots & \vdots & \vdots \\ g_1(x^{(n)}) & \cdots & g_p(x^{(n)}) \end{pmatrix} \tag{106}$$

- We can now write the emulator equation (91) at all runs $x_D = (x^{(1)}, \ldots, x^{(n)})$ as one

matrix equation:

$$
\begin{aligned}
f(x) &= g(x)^T\beta \;+\; u(x) && (107)\\
\Rightarrow \quad \begin{pmatrix} f(x^{(1)}) \\ \vdots \\ f(x^{(n)}) \end{pmatrix} &= \begin{pmatrix} g(x^{(1)})^T\beta \;+\; u(x^{(1)}) \\ \vdots \\ g(x^{(n)})^T\beta \;+\; u(x^{(n)}) \end{pmatrix} && (108)\\
\Rightarrow \quad \begin{pmatrix} f(x^{(1)}) \\ \vdots \\ f(x^{(n)}) \end{pmatrix} &= \begin{pmatrix} g(x^{(1)})^T \\ \vdots \\ g(x^{(n)})^T \end{pmatrix}\beta \;+\; \begin{pmatrix} u(x^{(1)}) \\ \vdots \\ u(x^{(n)}) \end{pmatrix} && (109)\\
\Rightarrow \quad D &= X\beta \;+\; U && (110)
\end{aligned}
$$

which we shall use shortly.

- We can now directly adjust the expectation of the regression coefficients β by the runs D:

$$
\begin{aligned}
\mathrm{E}_D[\beta] &= \mathrm{E}[\beta] \;+\; \mathrm{Cov}[\beta, D]\mathrm{Var}[D]^{-1}(D - \mathrm{E}[D]) \\
&= \mu_\beta \;+\; \mathrm{Cov}[\beta, X\beta + U]\mathrm{Var}[X\beta + U]^{-1}(D - X\mathrm{E}[\beta]) \\
&= \mu_\beta \;+\; \mathrm{Var}[\beta]X^T(X\mathrm{Var}[\beta]X^T + \mathrm{Var}[U])^{-1}(D - X\mu_\beta), \qquad \text{as } \mathrm{Cov}[\beta, U] = 0 \\
&= \mu_\beta \;+\; \Sigma_\beta X^T(X\Sigma_\beta X^T + \Omega)^{-1}(D - X\mu_\beta) && (111)\\
&= \mu_\beta \;+\; (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}X^T\Omega^{-1}(D - X\mu_\beta), \qquad \text{by MI2} \\
&= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}\left[(X^T\Omega^{-1}X + \Sigma_\beta^{-1})\mu_\beta + X^T\Omega^{-1}(D - X\mu_\beta)\right] \\
&= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}\left[\cancel{X^T\Omega^{-1}X\mu_\beta} + \Sigma_\beta^{-1}\mu_\beta + X^T\Omega^{-1}D - \cancel{X^T\Omega^{-1}X\mu_\beta}\right] \\
&= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}\left[\Sigma_\beta^{-1}\mu_\beta \;+\; X^T\Omega^{-1}D\right] \\
&= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}\left[\Sigma_\beta^{-1}\mu_\beta \;+\; (X^T\Omega^{-1}X)\,\hat{\beta}_{GLS}\right] && (112)\\
&\qquad\qquad \text{with } \hat{\beta}_{GLS} = (X^T\Omega^{-1}X)^{-1}X^T\Omega^{-1}D
\end{aligned}
$$

where we have used β_{GLS} which is the frequentist GLS estimate of regression coefficients β with design X, runs D and correlated error matrix Ω.

- While the expression for $\mathrm{E}_D[\beta]$ may not at first seem that intuitive, closer examination reveals clear insights: we see that the result is a (matrix) weighted sum of our prior expectation for β i.e. μ_β and the classical frequentist estimate for β, based on the run data alone, $\hat{\beta}_{GLS}$. The "weighting" is moderated by the prior precision matrix Σ_β^{-1} and the $(X^T\Omega^{-1}X)$ matrix which we recall(!) is the precision of the frequentist estimator for $\hat{\beta}_{GLS}$ as (in a classical frequentist variance of estimator sense) we have that $\mathrm{Var}[\hat{\beta}_{GLS}] = (X^T\Omega^{-1}X)^{-1}$.

- Hence, the surer we are about our prior, the closer $\mathrm{E}_D[\beta]$ will be to the prior μ_β and the less sure we are about the prior, the closer $\mathrm{E}_D[\beta]$ will be to the GLS estimate $\hat{\beta}_{GLS}$.

- Similarly we can adjust the variance of the regression coefficients β by the runs D:

$$\begin{aligned}
\mathrm{Var}_D[\beta] &= \mathrm{Var}[\beta] - \mathrm{Cov}[\beta, D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, \beta] \\
&= \Sigma_\beta - \mathrm{Cov}[\beta, X\beta + U]\mathrm{Var}[X\beta + U]^{-1}\mathrm{Cov}[X\beta + U, \beta] \\
&= \Sigma_\beta - \mathrm{Var}[\beta]X^T(X\mathrm{Var}[\beta]X^T + \mathrm{Var}[U])^{-1}X\mathrm{Var}[\beta], \qquad \text{as } \mathrm{Cov}[\beta, U] = 0 \\
&= \Sigma_\beta - \Sigma_\beta X^T(X\Sigma_\beta X^T + \Omega)^{-1}X\Sigma_\beta \qquad (113) \\
&= \Sigma_\beta - (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}X^T\Omega^{-1}X\Sigma_\beta, \qquad \text{by MI2} \\
&= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}\left[(X^T\Omega^{-1}X + \Sigma_\beta^{-1})\Sigma_\beta - X^T\Omega^{-1}X\Sigma_\beta\right] \\
&= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}\left[\cancel{X^T\Omega^{-1}X\Sigma_\beta} + I - \cancel{X^T\Omega^{-1}X\Sigma_\beta}\right] \\
&= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1} \qquad (114)
\end{aligned}$$

- We now see that $\mathrm{Var}_D[\beta]$ is formed from a combination of the precisions from the prior Σ_β^{-1} and the GLS estimator variance $\mathrm{Var}[\hat{\beta}_{GLS}]$.

- If we are highly uncertain regarding our prior specification for β we may be interested in examining the vague prior limit $\Sigma_\beta \to \infty$ or, more precisely, as Σ_β is a covariance matrix, the limit in which each eigenvalue of Σ_β tends to infinity. This is hard to perform in the original BL update form of $\mathrm{E}_D[f(x)]$ (see e.g. equation (111)), but is trivial in the weighted average form that we derived in equation (112), as we can easily impose the equivalent limit of $\Sigma_\beta^{-1} \to 0$.

- In the vague prior limit we therefore recover:

$$\begin{aligned}
\lim_{\Sigma_\beta^{-1}\to 0} \mathrm{E}_D[\beta] &= (X^T\Omega^{-1}X + 0)^{-1}\left[0 + (X^T\Omega^{-1}X)\,\hat{\beta}_{GLS}\right] = \hat{\beta}_{GLS} \\
\lim_{\Sigma_\beta^{-1}\to 0} \mathrm{Var}_D[\beta] &= (X^T\Omega^{-1}X + 0)^{-1} = (X^T\Omega^{-1}X)^{-1} = \mathrm{Var}[\hat{\beta}_{GLS}]
\end{aligned}$$

- Note that in the additional limit of far apart runs, or equivalently $\theta \to 0$ we have $\Omega \to \sigma^2 I$, where I is the $n \times n$ identity matrix and hence trivially,

$$\begin{aligned}
\lim_{\Omega\to\sigma^2 I}\left[\lim_{\Sigma_\beta^{-1}\to 0}\mathrm{E}_D[\beta]\right] &= \lim_{\Omega\to\sigma^2 I}\hat{\beta}_{GLS} = \lim_{\Omega\to\sigma^2 I}(X^T\Omega^{-1}X)^{-1}X^T\Omega^{-1}D \\
&= (X^T(\sigma^2 I)^{-1}X)^{-1}X^T(\sigma^2 I)^{-1}D \\
&= (X^TX)^{-1}X^TD = \hat{\beta}_{OLS} \\
\lim_{\Omega\to\sigma^2 I}\left[\lim_{\Sigma_\beta^{-1}\to 0}\mathrm{Var}_D[\beta]\right] &= \lim_{\Omega\to\sigma^2 I}\mathrm{Var}[\hat{\beta}_{GLS}] = \lim_{\Omega\to\sigma^2 I}(X^T\Omega^{-1}X)^{-1} \\
&= (X^T(\sigma^2 I)^{-1}X)^{-1} = \sigma^2(X^TX)^{-1} = \mathrm{Var}[\hat{\beta}_{OLS}]
\end{aligned}$$

hence we regain the Ordinary Least Squares estimates of standard linear regression.

- Alternatively, in the limit where we are very sure of our prior specification for β we of course trivially obtain by applying $\Sigma_\beta \to 0$ and using equations (111) and (113):

$$\begin{aligned}
\lim_{\Sigma_\beta\to 0}\mathrm{E}_D[\beta] &= \mu_\beta \\
\lim_{\Sigma_\beta\to 0}\mathrm{Var}_D[\beta] &= 0
\end{aligned}$$

which means our prior is dominating the runs data.

Extremely Useful Matrix Identities part 2

- Now we will prove a version of the celebrated "Sherman-Morrison-Woodbury Identity".

Theorem: Matrix Identity 3 (**MI3**)

- For square matrix P:

$$(I+P)^{-1} = I-(I+P)^{-1}P \tag{115}$$

Proof: Matrix Identity 3 (**MI3**)

$$\begin{aligned}
(I+P)^{-1} &= (I+P)^{-1}(I+P-P) \\
&= (I+P)^{-1}(I+P)-(I+P)^{-1}P \\
&= I-(I+P)^{-1}P
\end{aligned}$$

Theorem: Matrix Identity 4 (**MI4**) The "Sherman-Morrison-Woodbury Identity".

- For invertible matrices A and C, and matrices B and D of appropriate dimension:

$$(A+BCD)^{-1} \;=\; A^{-1} \;-\; A^{-1}B\,(C^{-1}+DA^{-1}B)^{-1}DA^{-1} \tag{116}$$

Proof: Matrix Identity 4 (**MI4**)

$$\begin{aligned}
(A+BCD)^{-1} &= (A\left[I+A^{-1}BCD\right])^{-1} \\
&= (I+A^{-1}BCD)^{-1}A^{-1} \\
&= \left[I-(I+A^{-1}BCD)^{-1}A^{-1}BCD\right]A^{-1}, && \text{by MI3} \\
&= A^{-1}-(I+A^{-1}BCD)^{-1}A^{-1}BCDA^{-1} \\
&= A^{-1}-A^{-1}B(I+CDA^{-1}B)^{-1}CDA^{-1}, && \text{by MI1} \\
&= A^{-1}-A^{-1}B\left[C^{-1}(I+CDA^{-1}B)\right]^{-1}DA^{-1} \\
&= A^{-1}-A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}
\end{aligned}$$

- We will use this shortly.

Adjusted Expectation and Variance of $u(x)$: $\mathrm{E}_D[u(x)]$ and $\mathrm{Var}_D[u(x)]$

- Remember that we want to find

$$\mathrm{E}_D[f(x)] \;=\; \mathrm{E}_D[g(x)^T\beta \;+\; u(x)] \;=\; g(x)^T\mathrm{E}_D[\beta] \;+\; \mathrm{E}_D[u(x)] \tag{117}$$

we have $\mathrm{E}_D[\beta]$ so now we turn our attention to $\mathrm{E}_D[u(x)]$. Defining the vector of covariances $a(x)^T = \mathrm{Cov}[u(x), U]$ we have:

$$\begin{aligned}
\mathrm{E}_D[u(x)] &= \mathrm{E}[u(x)] \;+\; \mathrm{Cov}[u(x),D]\mathrm{Var}[D]^{-1}(D-\mathrm{E}[D]) \\
&= 0 \;+\; \mathrm{Cov}[u(x),X\beta+U]\mathrm{Var}[X\beta+U]^{-1}(D-X\mu_\beta) \\
&= \mathrm{Cov}[u(x),U](X\Sigma_\beta X^T+\Omega)^{-1}(D-X\mu_\beta) \\
&= a(x)^T\left[(X\Sigma_\beta X^T+\Omega)^{-1}D-(X\Sigma_\beta X^T+\Omega)^{-1}X\Sigma_\beta\Sigma_\beta^{-1}\mu_\beta\right] \\
\text{using } \mathbf{MI2} \qquad & (X\Sigma_\beta X^T+\Omega)^{-1}X\Sigma_\beta \;=\; \Omega^{-1}X(X^T\Omega^{-1}X+\Sigma_\beta^{-1})^{-1} \\
&= a(x)^T\left[(X\Sigma_\beta X^T+\Omega)^{-1}D-\Omega^{-1}X(X^T\Omega^{-1}X+\Sigma_\beta^{-1})^{-1}\Sigma_\beta^{-1}\mu_\beta\right] \\
\text{using } \mathbf{MI4} \qquad & (X\Sigma_\beta X^T+\Omega)^{-1} \;=\; \Omega^{-1}-\Omega^{-1}X(X^T\Omega^{-1}X+\Sigma_\beta^{-1})^{-1}X^T\Omega^{-1} \\
&= a(x)^T\left[\left(\Omega^{-1}-\Omega^{-1}X(X^T\Omega^{-1}X+\Sigma_\beta^{-1})^{-1}X^T\Omega^{-1}\right)D- \right. \\
&\quad \left. \Omega^{-1}X(X^T\Omega^{-1}X+\Sigma_\beta^{-1})^{-1}\Sigma_\beta^{-1}\mu_\beta\right] \\
&= a(x)^T\Omega^{-1}\left[D-X(X^T\Omega^{-1}X+\Sigma_\beta^{-1})^{-1}\left\{X^T\Omega^{-1}D+\Sigma_\beta^{-1}\mu_\beta\right\}\right] \\
&= a(x)^T\Omega^{-1}(D-X\mathrm{E}_D[\beta])
\end{aligned}$$

- a remarkable simplification which provides substantial insight: we see that $\mathrm{E}[u(x)]$ is effectively updated/adjusted by the difference between the runs D and the *adjusted regression surface* $X\mathrm{E}_D[\beta]$, instead of the difference between D and the prior regression surface $X\mu_\beta$ as in the original BL update. Comparing first and last lines (and rewriting in terms of U, D and β we have shown that:

$$\begin{aligned}\mathrm{E}_D[u(x)] &= \mathrm{Cov}[u(x), D]\mathrm{Var}[D]^{-1}(D - X\mathrm{E}[\beta]) \\ &= \mathrm{Cov}[u(x), U]\mathrm{Var}[U]^{-1}(D - X\mathrm{E}_D[\beta])\end{aligned}$$

- Which has a very similar form to the BL update used in the simple emulator, with now $D - X\mathrm{E}_D[\beta]$ taking the role of a direct measurement of the residuals U.

- We can examine $\mathrm{E}_D[u(x)]$ in the vague/certain prior limits and the uncorrelated standard linear regression limits as before.

- Similarly, we can adjust the variance of $u(x)$ by the runs D:

$$\begin{aligned}\mathrm{Var}_D[u(x)] &= \mathrm{Var}[u(x)] - \mathrm{Cov}[u(x), D]\mathrm{Var}[D]^{-1}\mathrm{Cov}[D, u(x)] \\ &= \sigma^2 - \mathrm{Cov}[u(x), X\beta + U]\mathrm{Var}[X\beta + U]^{-1}\mathrm{Cov}[X\beta + U, u(x)] \\ &= \sigma^2 - \mathrm{Cov}[u(x), U]\mathrm{Var}[X\beta + U]^{-1}\mathrm{Cov}[U, u(x)] \\ &= \sigma^2 - a(x)^T(X\Sigma_\beta X^T + \Omega)^{-1}a(x) \\ &= \sigma^2 - a(x)^T\left[\Omega^{-1} - \Omega^{-1}X(X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}X^T\Omega^{-1}\right]a(x) \qquad \text{using } \mathbf{MI4} \\ &= \sigma^2 - a(x)^T\Omega^{-1}a(x) + a(x)^T\Omega^{-1}X\mathrm{Var}_D[\beta]X^T\Omega^{-1}a(x)\end{aligned}$$

- Here the first two terms look just like those for the simple emulator (i.e. without regression terms), but now we have an additional term that depends on $\mathrm{Var}_D[\beta]$. This is intuitive, as our uncertainty about the residual process $u(x)$ will be affected by whether we are sure (e.g. with $\mathrm{Var}_D[\beta] = 0$) or unsure ($\mathrm{Var}_D[\beta] > 0$) about where the regression surface $X\mathrm{E}_D[\beta]$ actually lies.

Adjusted Covariance of $g(x)^T\beta$ and $u(x)$: $\mathrm{Cov}_D[g(x)^T\beta, u(x)]$

- We have the pieces to find $\mathrm{E}_D[f(x)]$ (we will look at this shortly), but we also wish to find $\mathrm{Var}_D[f(x)]$ which involves the following terms:

$$\begin{aligned}\mathrm{Var}_D[f(x)] &= \mathrm{Var}_D[g(x)^T\beta + u(x)] \\ &= g(x)^T\mathrm{Var}_D[\beta]g(x) \;+\; \mathrm{Var}_D[u(x)] \;+\; \mathrm{Cov}_D[g(x)^T\beta, u(x)] \;+\; \mathrm{Cov}_D[u(x), g(x)^T\beta] \\ &= g(x)^T\mathrm{Var}_D[\beta]g(x) \;+\; \mathrm{Var}_D[u(x)] \;+\; 2\mathrm{Cov}_D[g(x)^T\beta, u(x)]\end{aligned}$$

  as the two covariance terms are of course equal.

- We have already calculated the first two variance terms, so the final(!) calculation we need is to find the following adjusted covariance for which we use the BL covariance update (equation (44)), and recall that we a priori judged $\mathrm{Cov}[\beta, u(x)] = 0$. The full calculation is left as an exercise on the problem sheet, but it shows that:

$$\mathrm{Cov}_D[g(x)^T\beta, u(x)] \;=\; -\,g(x)^T\mathrm{Var}_D[\beta]X^T\Omega^{-1}a(x) \tag{118}$$

- This represents the covariance induced between the regression surface $g(x)^T\beta$ at point x and the residual process $u(x)$: this perfectly compensates our uncertainty about both $g(x)^T\beta$ and $u(x)$ so that their sum $f(x) = g(x)^T\beta + u(x)$ will still interpolate known runs perfectly (again, see problem sheet for proof).

Combining all Results: $\mathrm{E}_D[f(x)]$ **and** $\mathrm{Var}_D[f(x)]$

- Summarising the 5 results we have so far:

Summary of Adjusted Emulator Component Results

$$\begin{aligned}
\mathrm{E}_D[\beta] &= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1}\left[\Sigma_\beta^{-1}\mu_\beta \;+\; (X^T\Omega^{-1}X)\,\hat{\beta}_{GLS}\right] \\
\mathrm{Var}_D[\beta] &= (X^T\Omega^{-1}X + \Sigma_\beta^{-1})^{-1} \\
\mathrm{E}_D[u(x)] &= a(x)^T\Omega^{-1}(D - X\mathrm{E}_D[\beta]) \\
\mathrm{Var}_D[u(x)] &= \sigma^2 - a(x)^T\Omega^{-1}a(x) + a(x)^T\Omega^{-1}X\mathrm{Var}_D[\beta]X^T\Omega^{-1}a(x) \\
\mathrm{Cov}_D[g(x)^T\beta, u(x)] &= -\,g(x)^T\mathrm{Var}_D[\beta]X^T\Omega^{-1}a(x)
\end{aligned}$$

- We can now construct $\mathrm{E}_D[f(x)]$ and $\mathrm{Var}_D[f(x)]$:

Emulator Adjusted Expectation

$$\begin{aligned}
\mathrm{E}_D[f(x)] &= g(x)^T\mathrm{E}_D[\beta] \;+\; \mathrm{E}_D[u(x)] \\
&= g(x)^T\mathrm{E}_D[\beta] \;+\; a(x)^T\Omega^{-1}(D - X\mathrm{E}_D[\beta])
\end{aligned}$$

Emulator Adjusted Variance

$$\begin{aligned}
\mathrm{Var}_D[f(x)] &= g(x)^T\mathrm{Var}_D[\beta]g(x) \;+\; \mathrm{Var}_D[u(x)] \;+\; 2\mathrm{Cov}_D[g(x)^T\beta, u(x)] \\
&= g(x)^T\mathrm{Var}_D[\beta]g(x) \;+\; \sigma^2 - a(x)^T\Omega^{-1}a(x) \\
&\quad + a(x)^T\Omega^{-1}X\mathrm{Var}_D[\beta]X^T\Omega^{-1}a(x) - 2g(x)^T\mathrm{Var}_D[\beta]X^T\Omega^{-1}a(x) \\
&= (g(x) - X^T\Omega^{-1}a(x))^T\mathrm{Var}_D[\beta](g(x) - X^T\Omega^{-1}a(x)) + \sigma^2 - a(x)^T\Omega^{-1}a(x) \\
&= g_A(x)^T\mathrm{Var}_D[\beta]g_A(x) \;+\; \sigma^2 \;-\; a(x)^T\Omega^{-1}a(x), \\
&\qquad\qquad \text{with } \; g_A(x) \;=\; (g(x) - X^T\Omega^{-1}a(x))
\end{aligned}$$

- These results allow us a powerful extension to our simple emulator from before: now we can include global structure and physical insight via the regression terms.
- Often, a substantial proportion of the computer model's behaviour can be captured using the global regression terms, especially when aided by expert scientific insight, leaving the weakly stationary process $u(x)$ free to capture the more local, complex and perhaps unexpected behaviour of the deviations of the function $f(x)$ from the regression surface $g(x)^T\mathrm{E}_D[\beta]$.

5.5 Advanced Emulator Part 2: Active and Inactive Inputs

- A critical advance in emulator construction when dealing with higher dimensional input and output spaces is the concept of *active* and *inactive inputs*.
- Often, physical computer models $f(x)$ may have a large number of input parameters x_k, $k = 1, \ldots, m$ and model outputs $f_i(x)$, $i = 1, \ldots, q$.
- However, for an individual output $f_i(x)$, there may only be a subset of the inputs that have a substantial impact on $f_i(x)$, with the remaining inputs hardly affecting $f_i(x)$ at all.
- We call the first group *active inputs* and donate them as x^A and the second group *inactive inputs*.
- We really want the structured parts of our emulator i.e. the regression terms $g(x^A)^T\beta$ and the weakly stationary process terms $u(x^A)$ to only depend on the active inputs x^A, as this *dimensional reduction* will make our emulator much more powerful.
- Ignoring the inactive inputs all together is an option, but this is a little dangerous as they will have some effect on $f_i(x)$ e.g. altering its output by say 3%, so we represent them via an unstructured nugget, represented as $w(x)$, which captures this uncertainty.
- For univariate output $f(x)$, our emulator equation now becomes:

$$f(x) \;=\; \sum_{j=1}^{p} \beta_j g_j(x^A) \;+\; u(x^A) \;+\; w(x) \tag{119}$$

  where now the basis functions $g(x^A)$ only depend on the active inputs, as does the weakly stationary process $u(x^A)$ which has typical covariance structure:

$$\mathrm{Cov}[u(x^A), u(x^{A\prime})] \;=\; \sigma^2 c(x^A - x^{A\prime}) \tag{120}$$

  while the nugget $w(x)$ is unstructured and has covariance structure:

$$\mathrm{Cov}[w(x), w(x')] = \begin{cases} \sigma_w^2 & \text{if} \quad x = x' \\ 0 & \text{otherwise} \end{cases} \tag{121}$$

- Often we set $\sigma_w^2 = \delta\sigma^2$ for some small value of δ e.g. 0.03 (corresponding to 3% of variance caused by the inactive variables), or we estimate it from the data (see later).

- Note that this leads to the combined covariance structure for $u(x^A) + w(x)$ taking a similar form to that of equation (61), e.g. assuming a Squared Exponential structure for $u(x^A)$ leads to,

$$\text{Cov}[u(x^A) + w(x), u(x^{A\prime}) + w(x')] \;=\; \sigma^2 \left[\exp\left\{-\frac{||x^A - x^{A\prime}||^2}{\theta^2}\right\} \;+\; \delta\, \mathbb{1}_{x=x'}\right] \tag{122}$$

- We note that, extremely helpfully, we can still use all the results derived in section 5.4 (for $\text{E}_D[\beta], \text{E}_D[u(x)]$, $\text{Var}_D[\beta]$, $\text{Var}_D[u(x)]$ etc.) by simply redefining the old $u(x) \to u(x^A) + w(x)$ and redefining $\sigma^2 c(x - x')$ to be given by the right hand side of equation (122).

Multivariate Outputs $f_i(x)$.

- For multivariate output $f_i(x)$, $i = 1, \ldots, q$ the emulator equation generalises to:

$$f_i(x) \;=\; \sum_{j=1}^{p_i} \beta_{ij} g_{ij}(x^{A_i}) \;+\; u_i(x^{A_i}) \;+\; w_i(x) \tag{123}$$

  where notably we have different sets of active inputs x^{A_i} per output; a different number p_i of different basis functions $g_{ij}(x^{A_i})$ per output, and a different $u_i(x^{A_i})$, $w_i(x)$ along with their hyperparameters $\sigma_i, \sigma_{w_i}, \theta_i$ per output.

- The dimensional reduction that use of active inputs allows is very powerful, as emulating in lower dimensions is far easier.

- The use of *different sets of active inputs per output* is extremely powerful and allows us to divide and conquer many seemingly complex computer models.

Selecting Basis Functions $g_{ij}(x^{A_i})$

- There are many ways to select the set of active inputs x^{A_i} for each output $f_i(x)$ and then subsequently to select basis functions $g_{ij}(x^{A_i})$.

- Often we specify a list of possible forms for the basis based on scientific expertise (but often just use low order polynomials), then sub-select from this list.

- A simple but effective approach is to use classic model selection i.e. select based on simple linear model fits using AIC or BIC criteria e.g. using the lm() and step() functions in R.

- We can split the task into two by a) selecting for linear terms in the inputs first, to identify x^{A_i}, then b) select higher-order terms (or more complex terms) in those active inputs only, to identify $g_{ij}(x^{A_i})$. See [1] for details.

6 Gaussian Process Emulation

- We have shown that we can do a lot with second order Bayes Linear emulators, and there is a lot more one can do with them.
- However, sometimes it is useful to go further and to specify full (joint) probability distributions for all uncertain quantities related to the computer model output $f(x)$, instead of just expectations, variances and covariances.
- This comes at a cost: we must be able to justify these distributional assumptions/judgements using more rigorous diagnostics, otherwise the results of our analysis may be affected by the specifics of distributional forms that we do not really believe.
- Assuming this is all fine, the Gaussian Process is a natural way to do this.
- A Gaussian Process emulator works in a very similar way to a Bayes linear emulator: for simple emulators they share the same second order structure.
- We write for uncertain computer model function output $f(x)$:

$$f(x) \sim GP(\mu(.), \sigma^2 c(.,.)) \tag{124}$$

  to state that we represent $f(x)$ as a Gaussian Process with mean (i.e. expectation) $\mu(x)$ and covariance structure $\sigma^2 c(x, x')$.
- But this is just giving it a name. What it really means is that for any finite number of n_P input points, we judge the collection of n_P outputs to have a joint normal distribution, with mean and covariance formed using $\mu(x)$ and $\sigma^2 c(x, x')$ (see below).
- E.g. for an unknown output $f(x)$ at input x we specify a priori a Normal distribution:

$$f(x) \sim N(\mu(x), \sigma^2) \tag{125}$$

  where similar to the BL case $\mu(x) \equiv \mathrm{E}[f(x)]$ and $\sigma^2 \equiv \mathrm{Var}[f(x)]$.
- The real magic is in the joint structure however: for a finite set of n_P inputs $x_P \equiv \{x^{(1)}, \ldots, x^{(n_P)}\}$ the vector of outputs $f(x_P) \equiv (f(x^{(1)}), \ldots, f(x^{(n_P)}))^T$ has multivariate Normal distribution with mean $\mu(x_P)$ and covariance $\Sigma(x_P)$, so

$$f(x_P) \sim N(\mu(x_P), \Sigma(x_P)) \tag{126}$$

  where

$$\mu(x_P) = \begin{pmatrix} \mu(x^{(1)}) \\ \vdots \\ \mu(x^{(n_P)}) \end{pmatrix} \quad \text{and} \quad \Sigma(x_P) = \sigma^2 \begin{pmatrix} c(x^{(1)}, x^{(1)}) & c(x^{(1)}, x^{(2)}) & \cdots & c(x^{(1)}, x^{(n_P)}) \\ c(x^{(2)}, x^{(1)}) & c(x^{(2)}, x^{(2)}) & \cdots & c(x^{(2)}, x^{(n_P)}) \\ \vdots & \vdots & \vdots & \vdots \\ c(x^{(n_P)}, x^{(1)}) & c(x^{(n_P)}, x^{(2)}) & \cdots & c(x^{(n_P)}, x^{(n_P)}) \end{pmatrix}$$
$$\equiv \sigma^2 C(x_P)$$

but this is just the same 2nd order structure as for our simple BL emulators. E.g. noticing that $\mu(x^{(i)}) \equiv \mathrm{E}[f(x^{(i)})]$ and if we set $x_D = x_P$ then $D = f(x_P)$ and hence $\mathrm{Var}[D] = \Sigma(x_P)$ etc.

Generating Realisations of a GP.

- However, now we have specified a joint prior *distribution* for $f(x_D)$, we can actually simulate *prior realisations* from the random vector $f(x_P)$.
- E.g. specifying $\mu(x) = 0$, $\sigma^2 = 0.5^2$ and squared exponential covariance $c(x, x') = \exp\{-|x - x'|^2/\theta^2\}$ with $\theta = 0.25$ and setting x_P to have $n_P = 201$ points:

$$x_P \;=\; (0, 0.005, 0.01, 0.015, \cdots, 0.99, 0.995, 1)$$

  we can simulate a single realisation of 201 values for the length 201 vector $f(x_P)$ and plot this against the inputs x_P. Figure 28a shows multiple such realisations with each one a different colour.
- Now we can see in more detail the effect of changing to different kinds of covariance function. Figure 28 also shows realisations from Matern $\nu = 3/2$, Matern $\nu = 5/2$ and Exponential Covariance functions described in section 2.6. Note that for the Exponential case the realisations seem spiky as they represent functions that are not even differentiable.

Conditioning on Runs D.

- So what happens when we observed a set of n runs $D = (f(x^{(1)}, \ldots, f(x^{(n)}))^T$? As we are using full distributions we should use Bayes Theorem to update our prior beliefs about the large set of n_P outputs $f(x_P)$, by conditioning on $D = f(x_D)$:

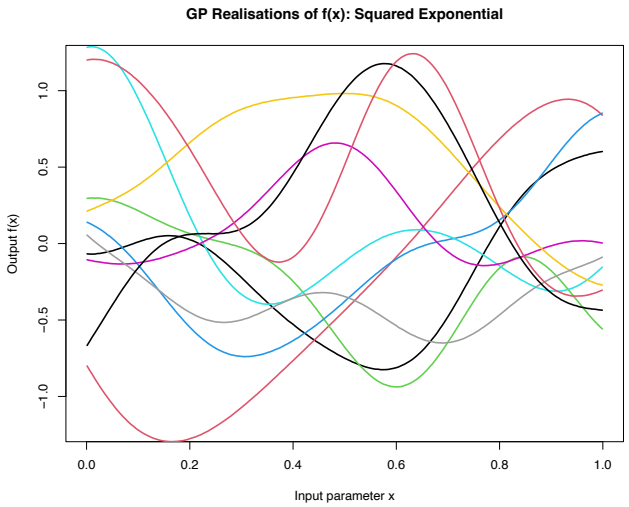
$$\pi(f(x_P)|D) \;=\; \frac{\pi(D|f(x_P))\pi(f(x_P))}{\pi(D)} \;=\; \frac{\pi(f(x_P), D)}{\pi(D)} \qquad (127)$$

  If fact it is actually easier to use the direct conditioning formula for the calculation, as given by the right hand side of equation (127).
- The updated posterior distribution for $f(x_P)|D$ is found to be also joint normal (due to the conditional behaviour of joint normals), but remarkably its second order structure also agrees precisely with our Bayes Linear update i.e. we have posterior

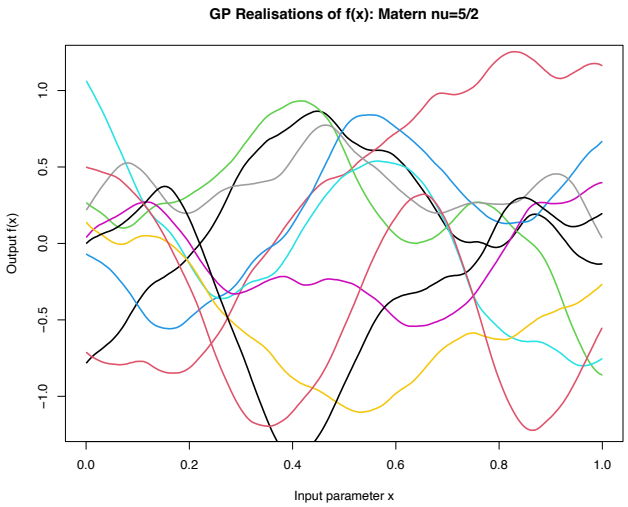
$$f(x_P)|D \;\sim\; N(\mu_D(x_P), \Sigma_D(x_P)) \qquad (128)$$

  where the conditioned moments are given by:

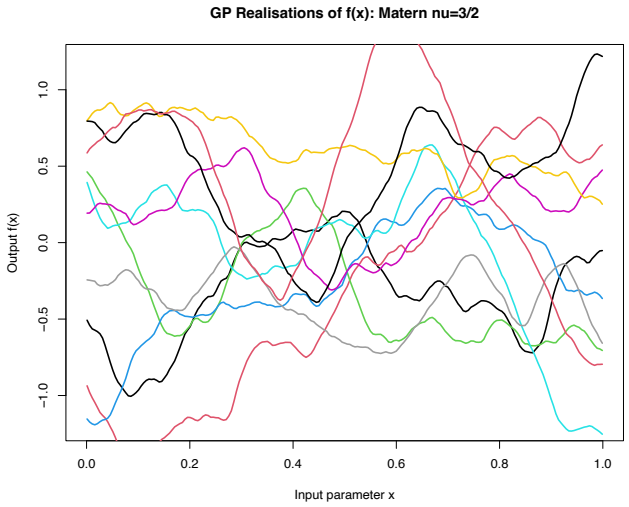
$$\mu_D(x_P) \;=\; \mathrm{E}_D[f(x_P)] \qquad \text{and} \qquad \Sigma_D(x_P) \;=\; \mathrm{Var}_D[f(x_P)] \qquad (129)$$



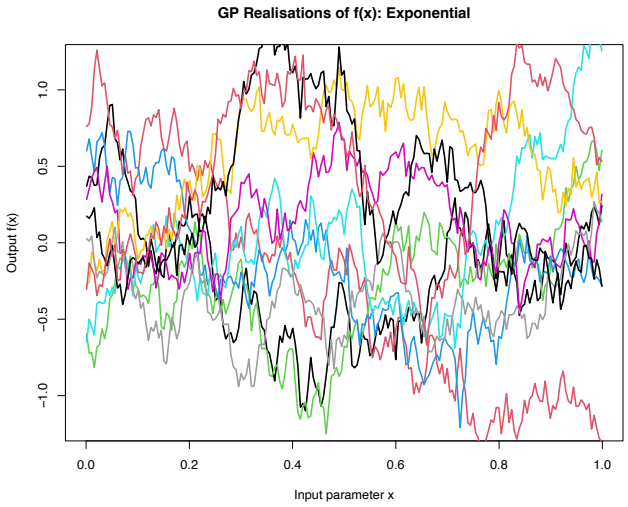
(a) Squared Exponential covariance structure.



(b) Matern $\nu = \frac{5}{2}$ covariance structure.



(c) Matern $\nu = \frac{3}{2}$ covariance structure.



(d) Exponential covariance structure.

Figure 28: 10 GP prior realisations of the function $f(x)$ using four different covariance structures. In each case $\theta = 0.25$ and $\sigma^2 = 0.5^2$.

where $\mathrm{E}_D[f(x_P)]$ is the length n_P vector of adjusted expectations and $\mathrm{Var}_D[f(x_P)]$ is the $n_P \times n_P$ adjusted variance of a random vector obtained from the full Bayes linear variance formula, given in section 2.3. The full proof of this is left to an exercise: it is not too hard but a bit messy and relies on certain inverse identities.

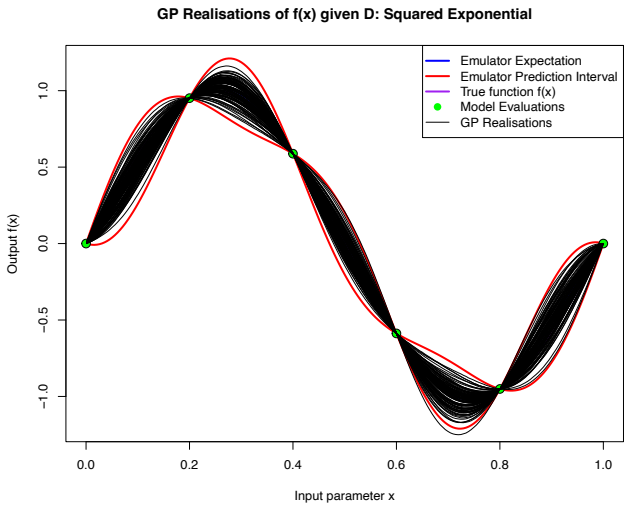
- We can now simulate realisations from the posterior distribution $f(x_P)|D$ as shown in Figure 29 for four choices of covariance function.

Estimating Emulator Hyperparameters via Maximum Likelihood

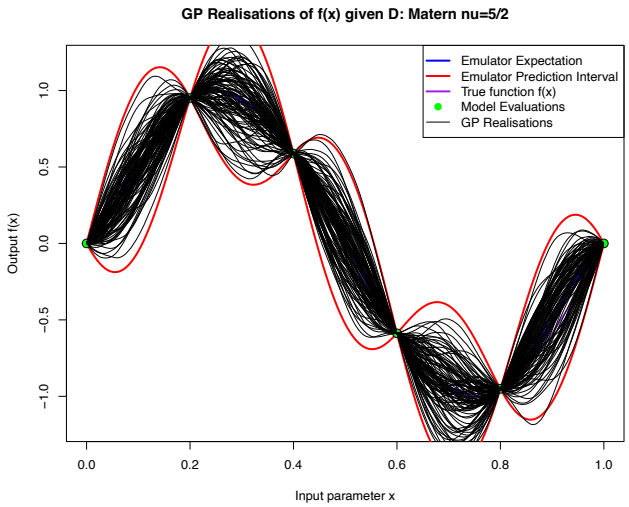
- Now we have specified distributions we can construct the likelihood for emulator parameters σ^2 and θ given a set of n runs D.
- We have a priori, inserting the conditioning on hyperparameters σ^2 and θ explicitly, and remembering that $D \equiv f(x_D)$, that the likelihood and log-likelihood are:

$$\begin{aligned} f(x_D)|\sigma^2,\theta &\sim N(\mu(x_D),\Sigma(x_D)) \\ \Rightarrow \quad \ell(\sigma^2,\theta) &= \pi(f(x_D)|\sigma^2,\theta) \\ &= \frac{1}{(2\pi)^{n/2}|\Sigma(x_D)|^{1/2}} \exp\Big\{-\big(D-\mu(x_D)\big)^T\Sigma(x_D)^{-1}\big(D-\mu(x_D)\big)\Big\} \\ \Rightarrow \quad L(\sigma^2,\theta) &= \log(\ell(\sigma^2,\theta)) \\ &= -\frac{n}{2}\log(2\pi) \;-\; \frac{1}{2}\log|\Sigma(x_D)| \;-\; \big(D-\mu(x_D)\big)^T\Sigma(x_D)^{-1}\big(D-\mu(x_D)\big) \end{aligned}$$

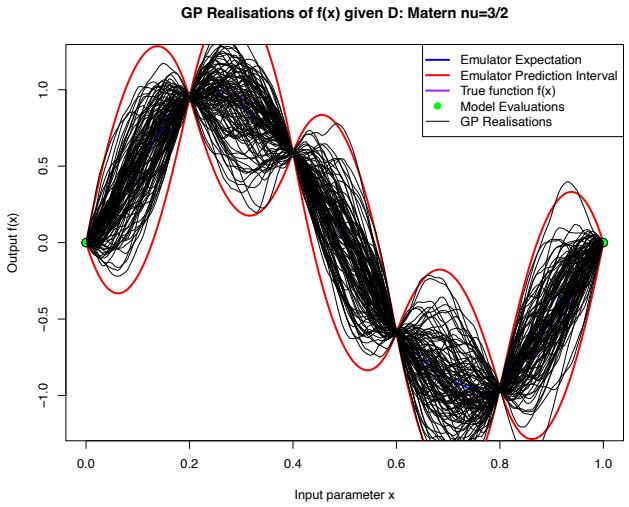
- If we write $\Sigma(x_D) = \sigma^2 C(x_D)$ as defined above, it is possible to find the maximum likelihood estimate $\hat{\sigma}^2_{MLE}$ by solving $\partial L(\sigma^2,\theta)/\partial\sigma^2 = 0$: see exercise sheet.
- Finding the maximum likelihood estimate for θ is not in general analytically possible, but we can find it numerically using e.g. the optim() function in R.
- These MLE's can be used as 'plug-in' values in our emulator, which can be OK (and is widely done), or we can treat them in a more sophisticated Bayesian way.
- There are dangers here: we should note that the MLE's and even the Bayesian equivalent, rely on the Normal distributional assumptions of the GP, which may be badly violated for a real computer model, hence making these estimates somewhat untrustworthy. Other, more robust methods are to use cross-validation (see e.g. [29]).
- The use of GPs facilitates full computer model *calibration* where we find the posterior distribution of x^*, see e.g. [30], which goes beyond HM, however, again this can be sensitive to the Normality assumptions in the GP.
- A major use of such GP emulators is in the area of Bayesian Optimisation, which we will explore next.



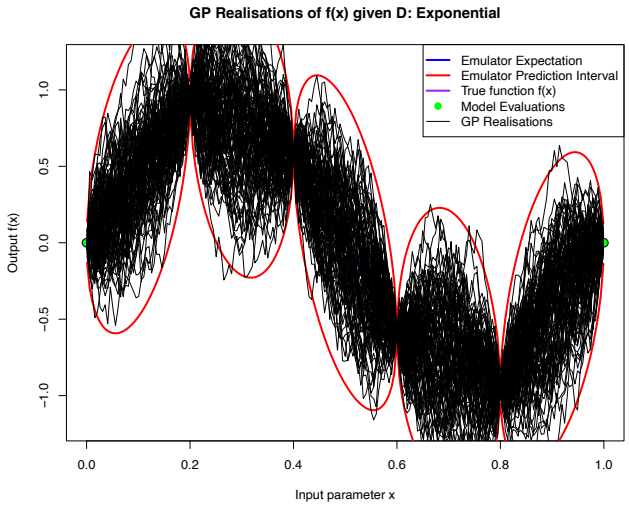
(a) Squared Exponential covariance structure.



(b) Matern $\nu = \frac{5}{2}$ covariance structure.



(c) Matern $\nu = \frac{3}{2}$ covariance structure.



(d) Exponential covariance structure.

Figure 29: 10 GP posterior realisations of the function $f(x)$ using four different covariance structures. In each case $\theta = 0.25$ and $\sigma^2 = 0.5^2$.

7 Bayesian Optimisation

- Often, the computer model output $f(x)$ may represent a (scalar) quantity which we wish to maximise e.g. (a) profit for some financial investment strategy, (b) number of people surviving a major pandemic, (c) long-term sustainable power output from a nuclear reactor etc.
- In a more sophisticated analysis $f(x)$ may represent our Utility of a set of future outcomes, defined in the Decision Theoretic sense, and x may represent a list of decision parameters that we can choose.
- E.g. for a pandemic our Utility may be formed from terms representing the number of people who survive, but also terms representing the damage caused by lockdowns to the economy, schooling and the social and domestic consequences of such actions, and also the costs of various medical interventions e.g. large scale rollout of vaccines, testing etc. The decision inputs x may represent the beginning and end point of lockdown, percentage coverage of vaccination program etc.
- Global warming is another complex example: forming a utility will be complicated and expensive to evaluate (as it will involve heavy climate models).
- In all these cases we would wish to find the decisions that maximise $f(x)$ i.e. we therefore wish to find (or to get very close to)

$$x_{max} = \underset{x \in \mathcal{X}}{\operatorname{argmax}} f(x) \tag{130}$$

  and hence find the corresponding $f_{max} = f(x_{max})$.
- As before, we assume that we have already evaluated the function at an initial set of n input locations $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ giving a column vector of model output values $D = (f(x^{(1)}), f(x^{(2)}), \ldots, f(x^{(n)}))^T$
- We denote the best/highest run output so far found as $f^+ = f(x^+)$ where the corresponding best input so far found is

$$x^+ = \underset{x^{(i)} \in \{x^{(1)}, \ldots, x^{(n)}\}}{\operatorname{argmax}} f(x^{(i)}) \tag{131}$$

- To maximise $f(x)$ there are various criteria discussed in the literature for choosing subsequent run locations: often runs are chosen one at a time.
- In many popular cases, these are myopic criteria that give the location of the next run $x^{(n+1)}$ using various "Acquisition Functions", as we now describe.
- Often these criteria rely on use of GP emulators instead of just BL emulators, as they need the actual distribution of unknown $f(x)$.

7.1 Acquisition Functions: Probability of Improvement $PI(x)$

- Here we summarise the classic Probability of Improvement criteria, to aid comparison of the strengths and weaknesses of it and of the related expected improvement criteria described in the next section.
- To decide on the location of the next run, $x^{(n+1)}$, this criteria suggests we find the point x that maximises the probability of improvement denoted $PI(x)$, that is the probability that $f(x)$ is greater than $f(x^+)$ the best output so far,

$$\begin{aligned} PI(x) &= P(f(x) \geq f(x^+)) && (132) \\ &= \Phi\left(\frac{\mu_D(x) - f(x^+)}{\sigma_D(x)}\right) && (133) \end{aligned}$$

  where $\Phi(.)$ is the cumulative distribution function of the standard normal distribution (as we use GP emulators here), and $\mu_D(x) = \mathrm{E}_D[f(x)]$ and $\sigma^2_D(x) = \mathrm{Var}_D[f(x)]$ are the emulator mean and variance updated by runs D defined previously.
- While initially looking intuitively advantageous (and having been heavily used in the past), this criteria has major problems: it seeks values of x that assure a high chance of improvement, even if the actual expected size of the improvement is extremely small.
- Hence it exhibits predominantly *exploitation* behaviour: locating the next run close to previous high runs to exploit knowledge of where the function is high.
- It is often said that a good acquisition function should choose new points that display a balance between *exploitation* and *exploration*, the latter being the choosing of runs in new, unexplored locations, where the emulator is currently uncertain. The $PI(x)$ criteria fails at this.
- An adaptation is to look at the probability of improvement higher than a minimum threshold $\xi > 0$ which alters the $PI(x)$ into $PI_\xi(x)$ defined as

$$\begin{aligned} PI_\xi(x) &= P(f(x) \geq f(x^+) + \xi) && (134) \\ &= \Phi\left(\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}\right) && (135) \end{aligned}$$

- While again seemingly sensible, this version is now very sensitive to the choice of ξ.
- Relaxation schemes have been considered, where ξ is initially set to a large value and then reduced, but again it can prove very hard to choose an appropriate problem dependent scheme.
- There is a fundamental structural problem with this type of criteria, that we will discuss more when we map it to a formal decision theory structure: see below.

7.2 Acquisition Functions: Expected Improvement $EI(x)$

- We now describe a much celebrated and widely used acquisition function known as the expected improvement and denoted $EI(x)$.

- This attempts to find a good balance between exploration and exploitation, by altering the core quantity of interest from the probability of (any) improvement as in the $PI(x)$ function, to the $EI(x)$ function, defined as the expectation of the linear improvement function $\mathcal{I}(x)$. Here $\mathcal{I}(x)$ is defined as

$$\mathcal{I}(x) \;=\; \max\{0, f(x) - f(x^+)\} \tag{136}$$

  We see that $\mathcal{I}(x)$ is positive when the new value of $f(x)$ is higher than the best value seen so far, and zero otherwise.

- The criteria instructs us to pick the new run location x that maximises the $EI(x)$ criteria, defined as the expectation of $\mathcal{I}(x)$

$$\begin{aligned} x &= \operatorname*{argmax}_x \mathrm{E}[\mathcal{I}(x)] && (137)\\ &= \operatorname*{argmax}_x \mathrm{E}[\max\{0, f(x) - f(x^+)\}] && (138) \end{aligned}$$

- We know the distribution of the uncertain $f(x)$ from the GP emulator, therefore we have that $f(x) \sim N(\mu_D(x), \sigma_D^2(x))$ where $\mu_D(x)$ and $\sigma_D^2(x)$ is the emulator mean and variance at input point x.

- We can then rewrite $\mathcal{I}(x)$ in terms of the random variable $Z = (f(x) - \mu_D(x))/\sigma_D(x)$ where $Z \sim N(0,1)$ which has standard normal distribution, so that

$$\begin{aligned} \mathcal{I}(x) &= \max\{0, f(x) - f(x^+)\} && (139)\\ &= \max\{0,\ f(x) - \mu_D(x) + \mu_D(x) - f(x^+)\} && (140)\\ &= \max\{0,\ \sigma_D(x)\, Z + \mu_D(x) - f(x^+)\} && (141)\\ &= \max\{0,\ \sigma_D\, Z + \mu_D - f^+\} && (142) \end{aligned}$$

  where in the last line we suppress the x dependence of μ_D and σ_D and use $f^+ = f(x^+)$, and note that the only random quantity here is Z. Hence we can use this to find the expectation of $\mathcal{I}(x)$ analytically:

$$
\begin{aligned}
\mathrm{E}[\mathcal{I}(x)] &= \int_{-\infty}^{\infty} \mathcal{I}(x)\, \pi(f|\mu_D, \sigma_D^2)\, df && (143)\\
&= \int_{f^+=f(x^+)}^{\infty} (f(x) - f^+)\, \pi(f|\mu_D, \sigma_D^2)\, df && (144)\\
&= \int_{z^+=\frac{f^+-\mu_D}{\sigma_D}}^{\infty} (\sigma_D z + \mu_D - f^+)\, \pi(z|0,1)\, dz && (145)\\
&= \sigma_D \int_{z^+}^{\infty} z\pi(z|0,1)dz \;+\; (\mu_D - f^+) \int_{z^+}^{\infty} \pi(z|0,1)\, dz && (146)\\
&= \sigma_D \int_{z^+}^{\infty} z \frac{e^{-\frac{1}{2}z^2}}{\sqrt{2\pi}}\, dz \;+\; (\mu_D - f^+)\, [\Phi(z)]_{z^+}^{\infty} && (147)\\
&= \sigma_D \left[\frac{-1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2}\right]_{z^+}^{\infty} \;+\; (\mu_D - f^+)\, [\Phi(z)]_{z^+}^{\infty} && (148)\\
&= \sigma_D \phi(z^+) \;+\; (\mu_D - f^+)\left[1 - \Phi(z^+)\right] && (149)\\
&= \sigma_D \phi(z^*) \;+\; (\mu_D - f^+)\Phi(z^*), && (150)
\end{aligned}
$$

where on the last line we have defined $z^* = -z^+ = (\mu_D - f^+)/\sigma_D$ and $\phi(.)$ denotes the probability density function of the standard normal distribution and $\Phi(.)$ denotes the cumulative distribution function of the standard normal distribution, as before.

- The expected improvement is much used mainly because a) it has an analytic expression, which speeds up the search for the next input x, and b) it displays both exploitation and exploration behaviour.

- The latter can be seen from equation (150): if z^* is large and positive, the second term on the rhs dominates and we favour points with large emulator mean μ_D, resulting in exploitation, whether if $|z^*|$ is small, the first term dominates and we favour points with large emulator variance σ_D^2, resulting in exploration.

- Having said this, it is unclear as to whether the relative amounts of exploitation and exploration, or the particular way they are employed is in any way optimal.

- Also note that although analytic expressions are of course beneficial, numerically evaluating expectations over 1-dimensional normal distributions are very fast too, leaving open the possibility of examining the expectation of more complex functions of $f(x)$.

Example: Bayesian Optimisation in 1D

- We now demonstrate the use of $EI(x)$ on a simple 1-dimensional example. We will also show the form of $PI(x)$ and highlight its deficiencies.
- In section 7.4 we will return to this example with a more exploration preferring criteria.
- We wish to emulate and optimise the 1-dimensional function

$$f(x) = \sin\left(\frac{6\pi}{10}x\right) + \frac{1}{20}x, \tag{151}$$

  over the interval $0 \leq x \leq 10$, which has deliberately been chosen to have multiple modes.
- We construct a simple zero mean Gaussian process emulator, with the correlation length $\theta = 1.3$, prior variance $\sigma_u^2 = 1.4^2$, zero regression terms so $\beta_{ij} = 0$ and a small nugget for numerical stability $\sigma_w^2 = 10^{-5}\sigma_u^2$. We assume two runs have already been performed at locations $x^{(1)} = 1$ and $x^{(2)} = 9$.
- Figure 30 shows the emulator of $f(x)$ and various acquisition functions, given the above two runs have been evaluated. The top row shows the emulator predictions: emulator mean $\mu_D(x) = \mathrm{E}_D[f(x)]$ (blue lines), prediction interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines) where the emulator standard deviation is $\sigma_D(x) = \sqrt{\mathrm{Var}_D[f(x)]}$, with the evaluated runs and their locations given by the black points/grey vertical lines. The middle row gives the $PI(x)$ criteria (red lines). The bottom row gives the $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4.
- Figures 30 to 44 show the development of the emulator and acquisition functions as we sequentially include more evaluations of $f(x)$. The three plots in the left column of each of these figures use the $EI(x)$ criteria to choose the next point (its location is shown by the blue vertical lines in the bottom row plots).
- From these figures we see several features of interest. We see that the $PI(x)$ criteria displays mostly exploitative behaviour, mostly preferring locations close to the known maximum run, which is undesirable in general.
- The $EI(x)$ criteria does better, balancing the pursuit of high values of $f(x)$ with some modest amount of exploration. However, there are several cases where we may prefer more exploration. For example, in figure 31 the next run is chosen to be at approximately $x = 1.9$, which is relatively close to the highest run at $x = 1$, however a better choice would be in a more uncertain, and hence informative, region e.g. at

$x = 5$. Similar behaviour is seen further into the optimisation in figure 34, where we may prefer to locate a run at $x = 4$ again to pin down a highly uncertain region. Finally, from figure 41 to 44 the $EI(x)$ focuses on the third peak (around the correct maximum at about $x = 7.6$), however we may wish it to rule out the first peak which is still somewhat uncertain, especially around $x = 0.8$.

- Overall, the $EI(x)$ criteria is still behaving well resulting in a good final emulator. The issues we see can escalate in higher dimensions however, as we shall see.

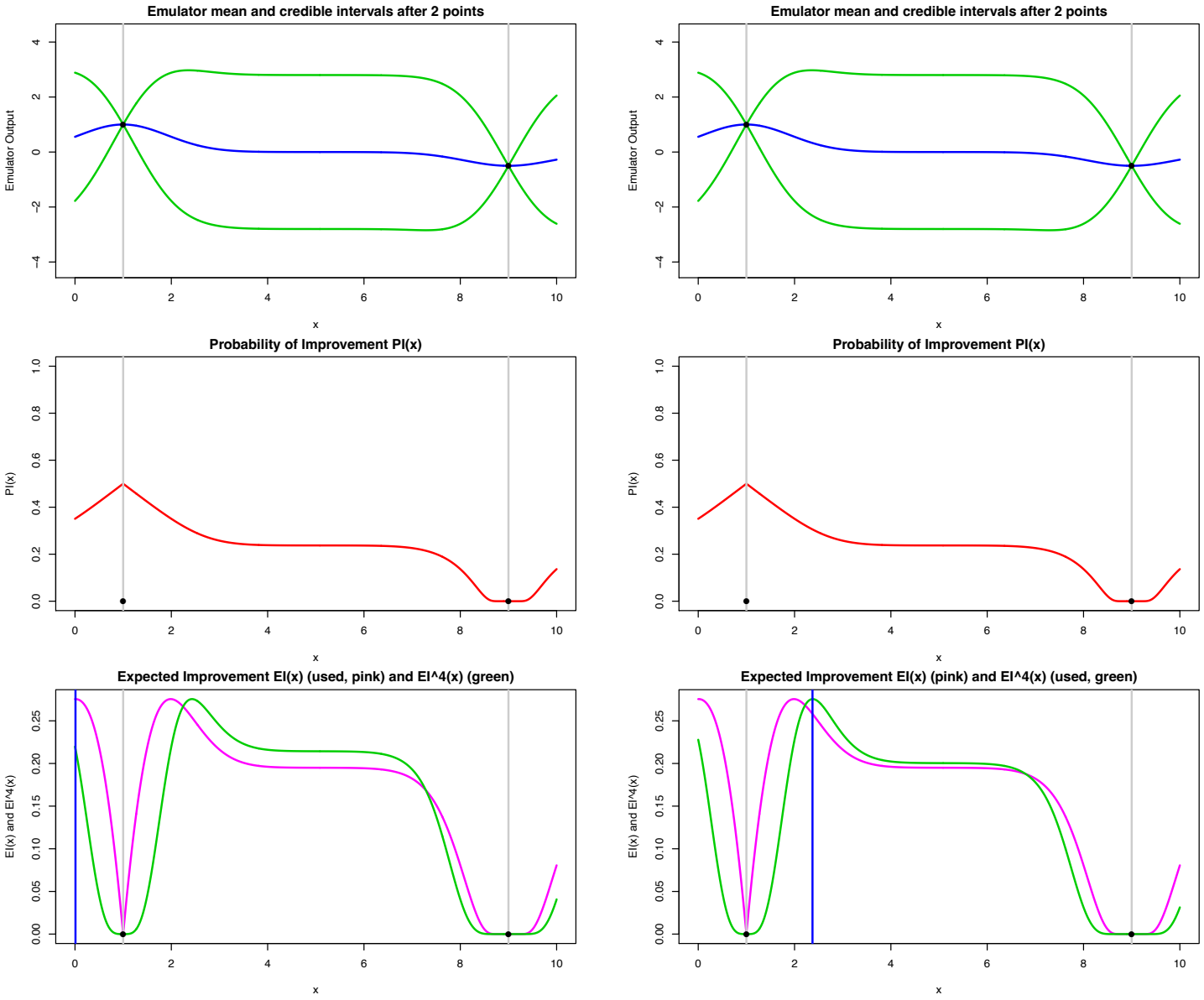


Figure 30: The 1-dimensional emulator of $f(x)$ after 2 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

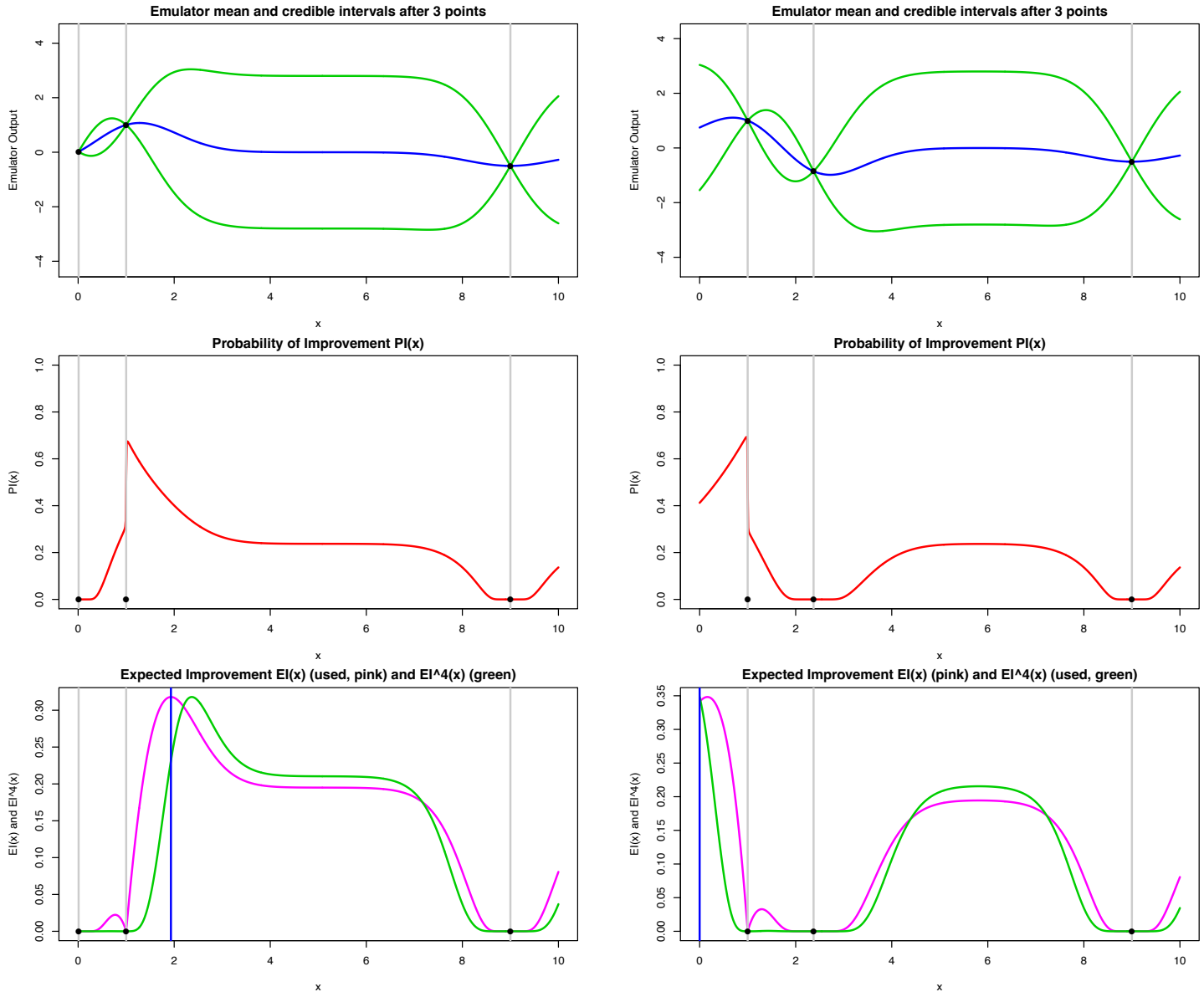


Figure 31: The 1-dimensional emulator of $f(x)$ after 3 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

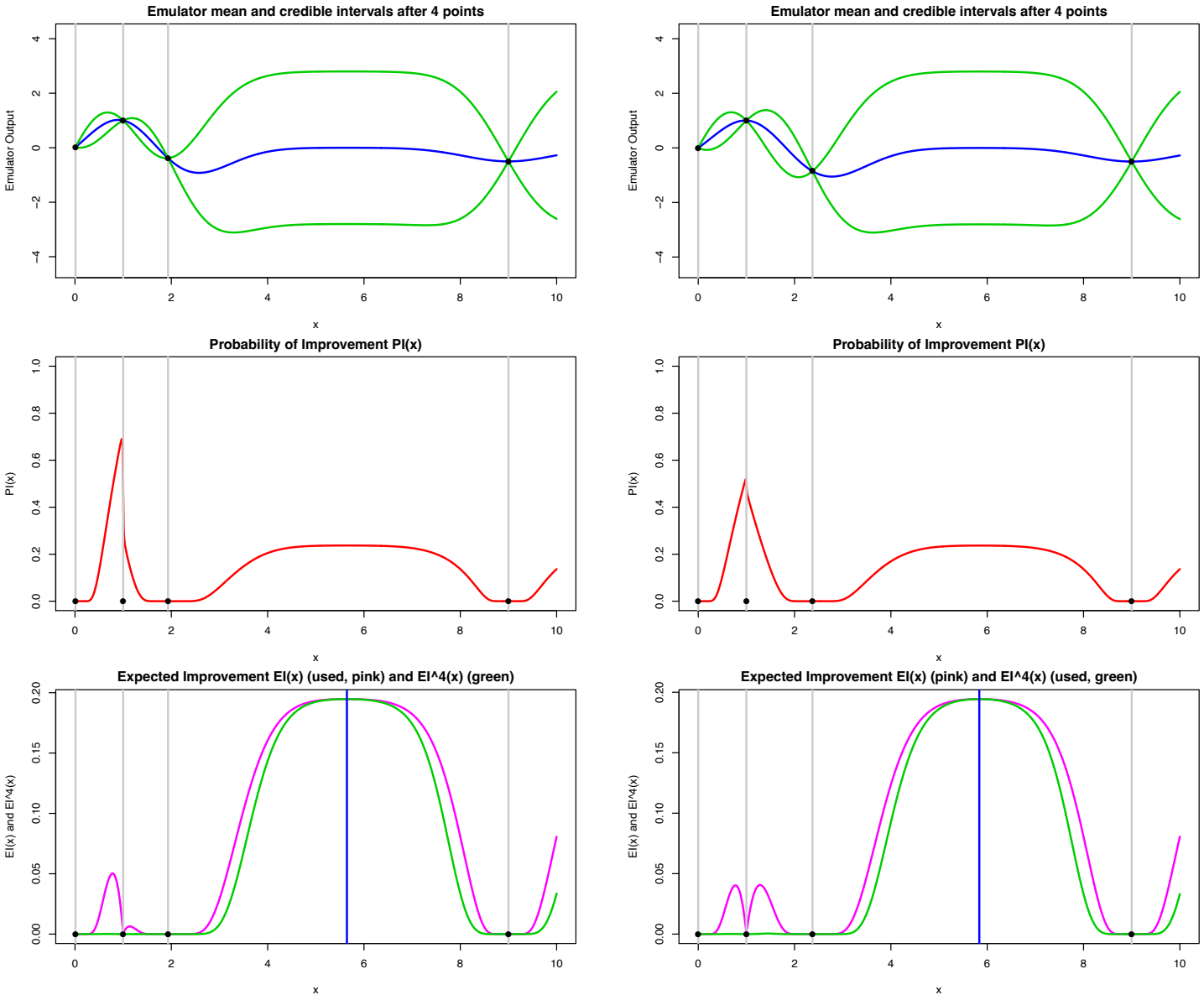


Figure 32: The 1-dimensional emulator of $f(x)$ after 4 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

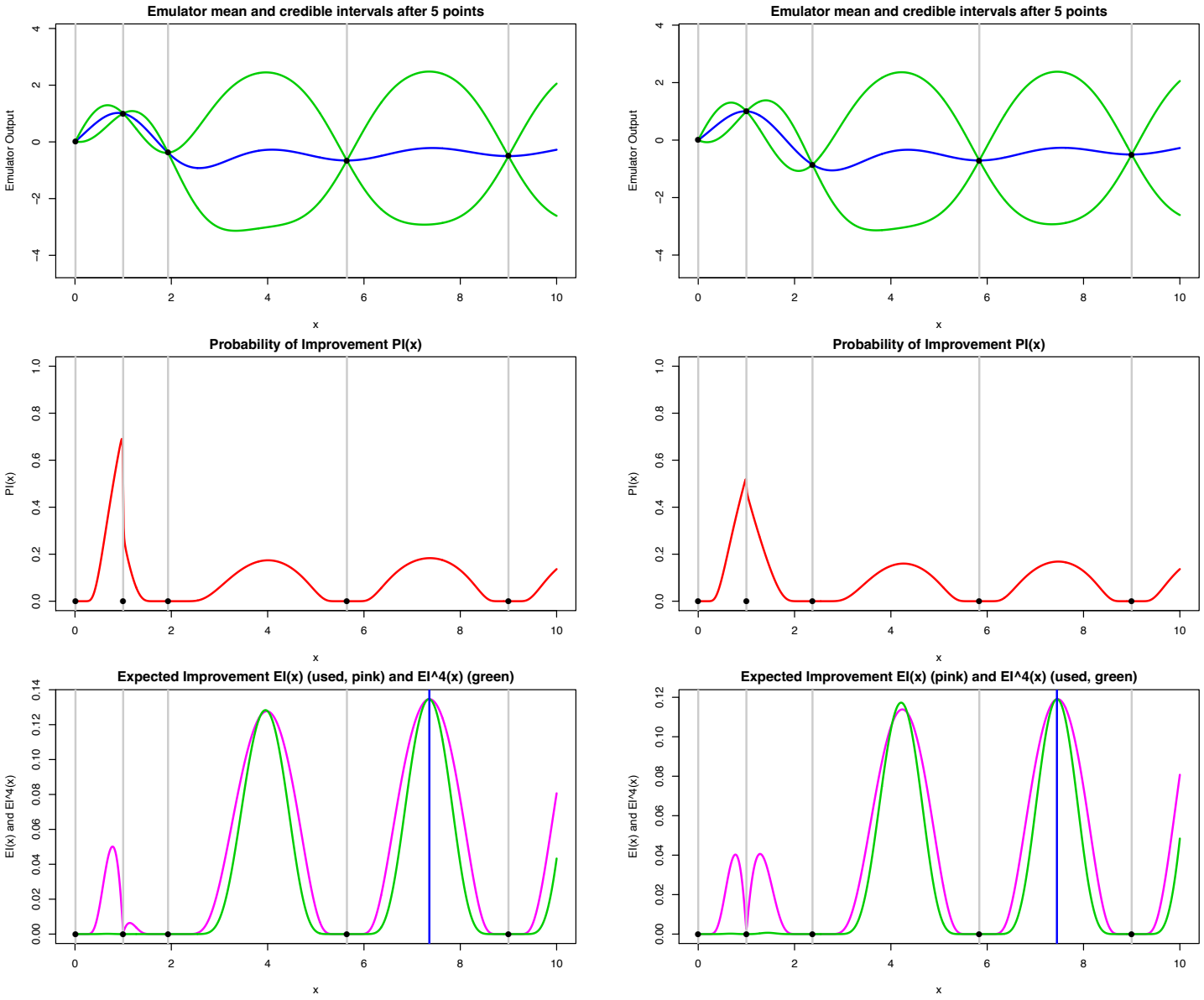


Figure 33: The 1-dimensional emulator of $f(x)$ after 5 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

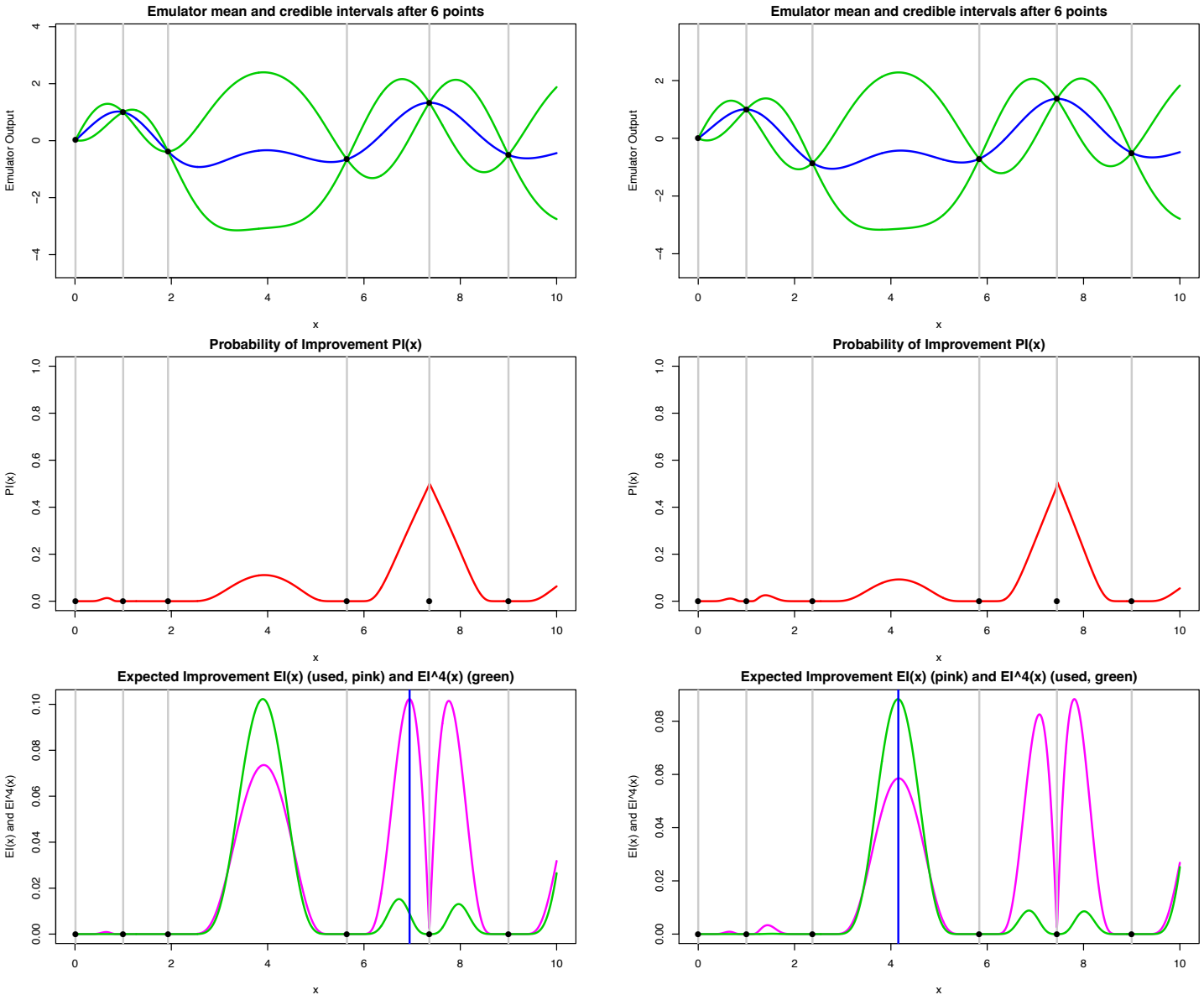


Figure 34: The 1-dimensional emulator of $f(x)$ after 6 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

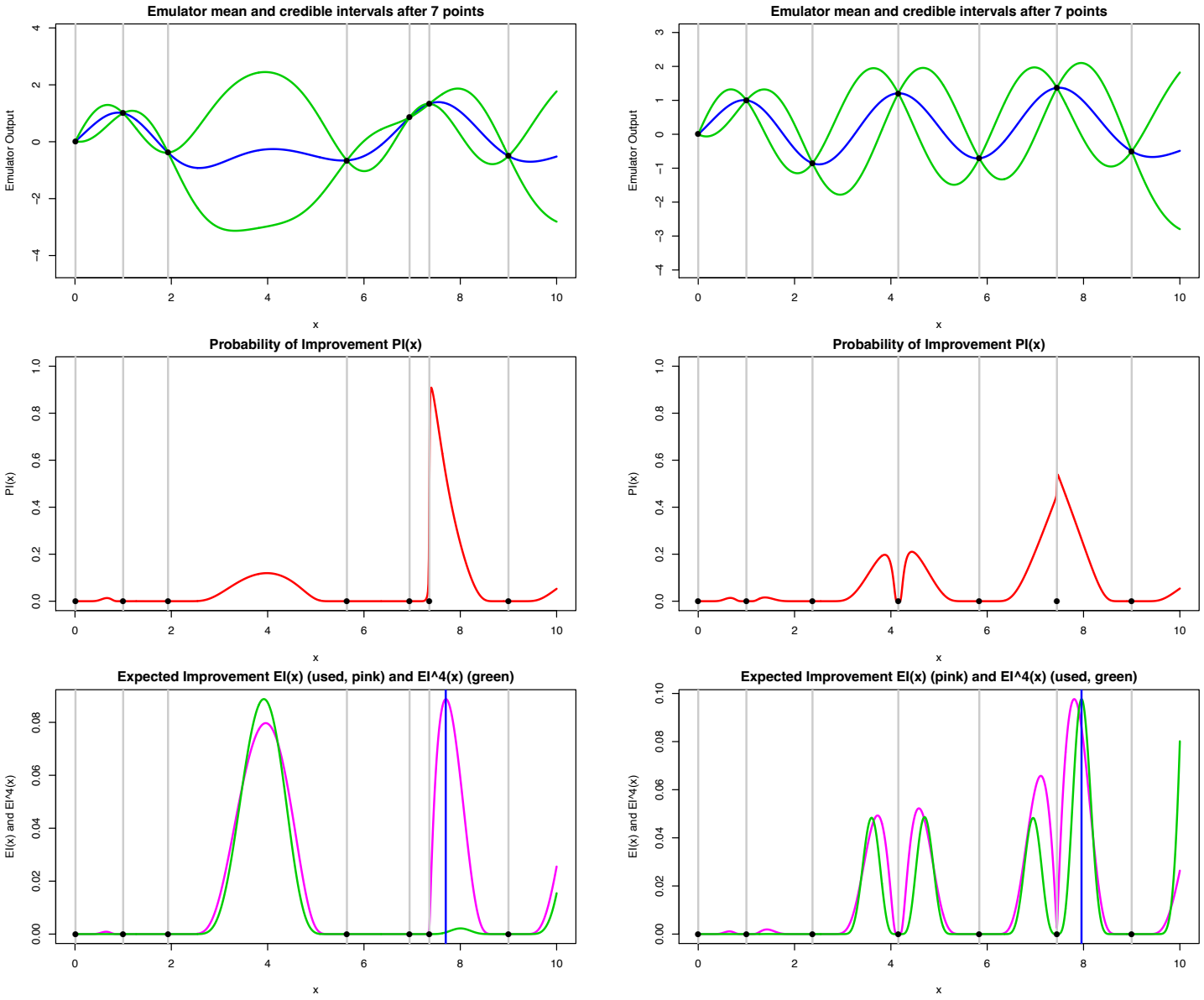


Figure 35: The 1-dimensional emulator of $f(x)$ after 7 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

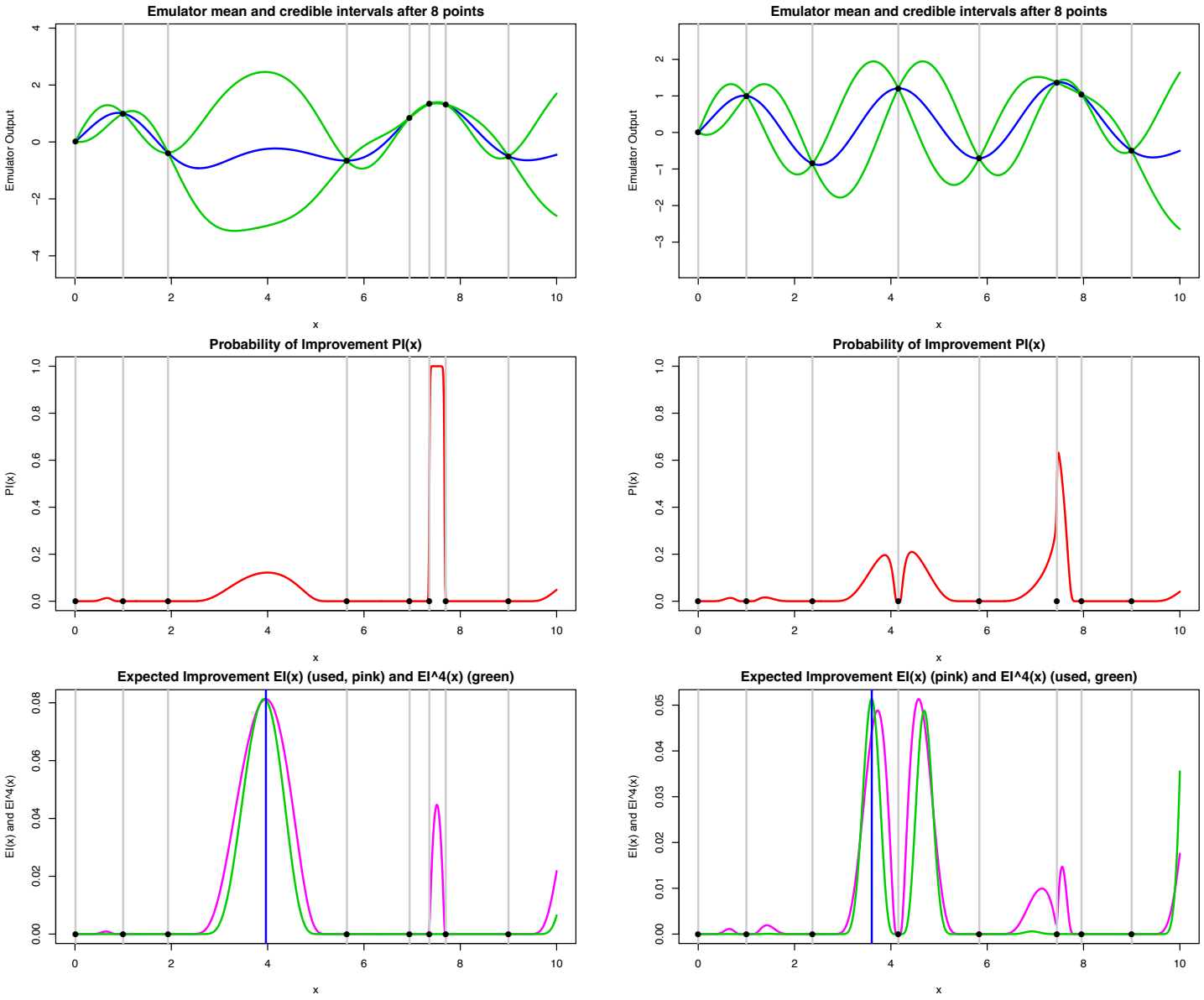


Figure 36: The 1-dimensional emulator of $f(x)$ after 8 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

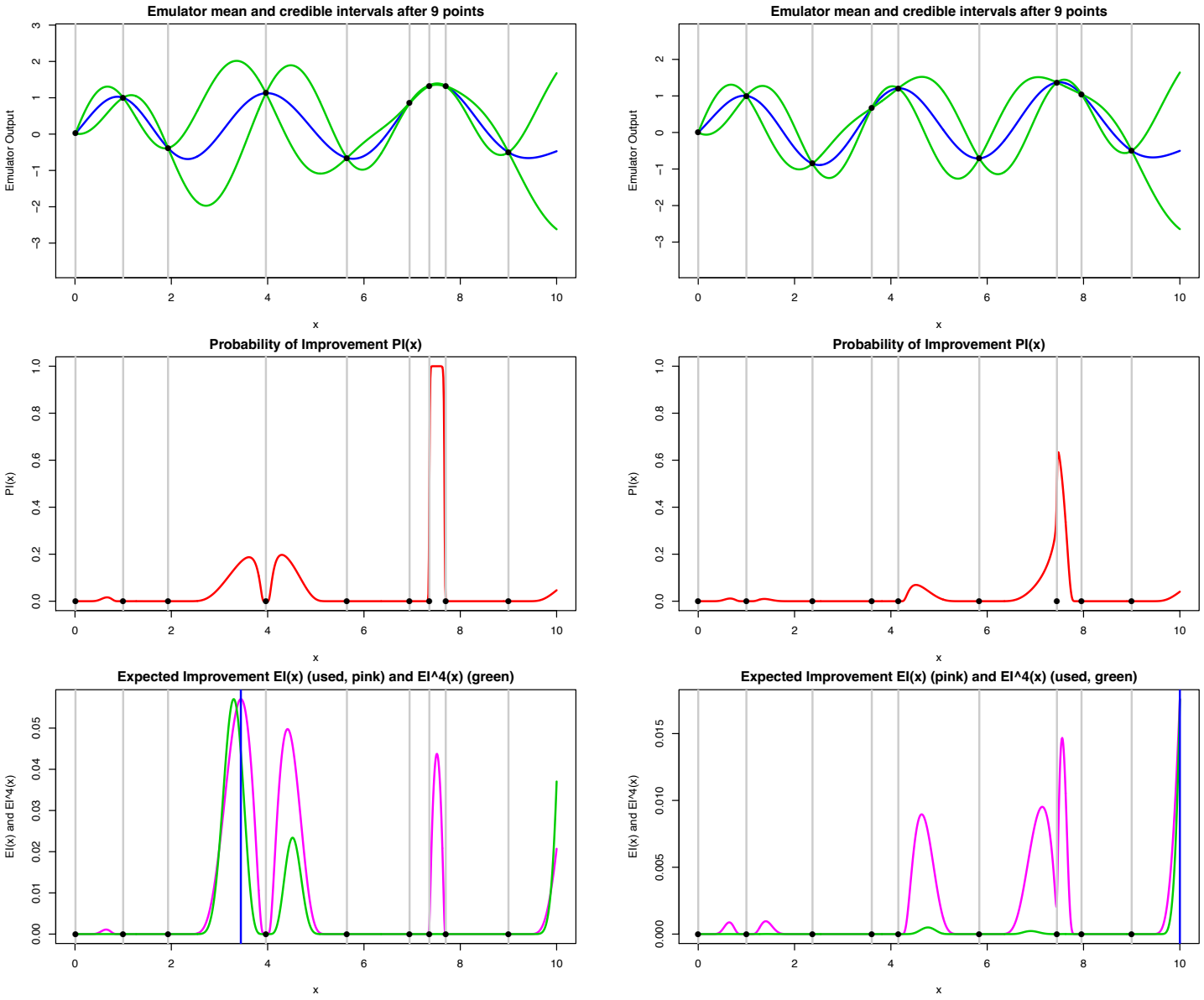


Figure 37: The 1-dimensional emulator of $f(x)$ after 9 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

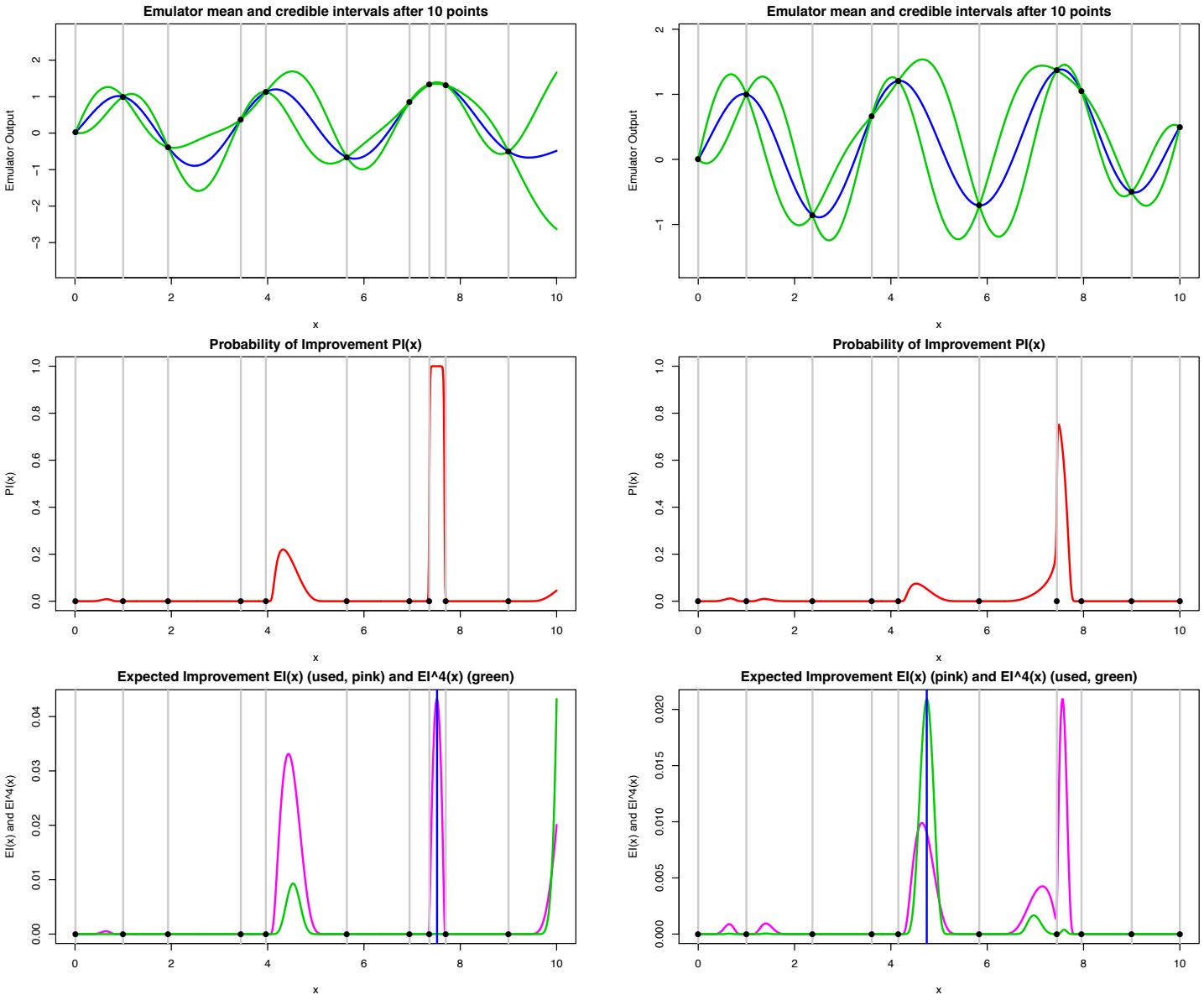


Figure 38: The 1-dimensional emulator of $f(x)$ after 10 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

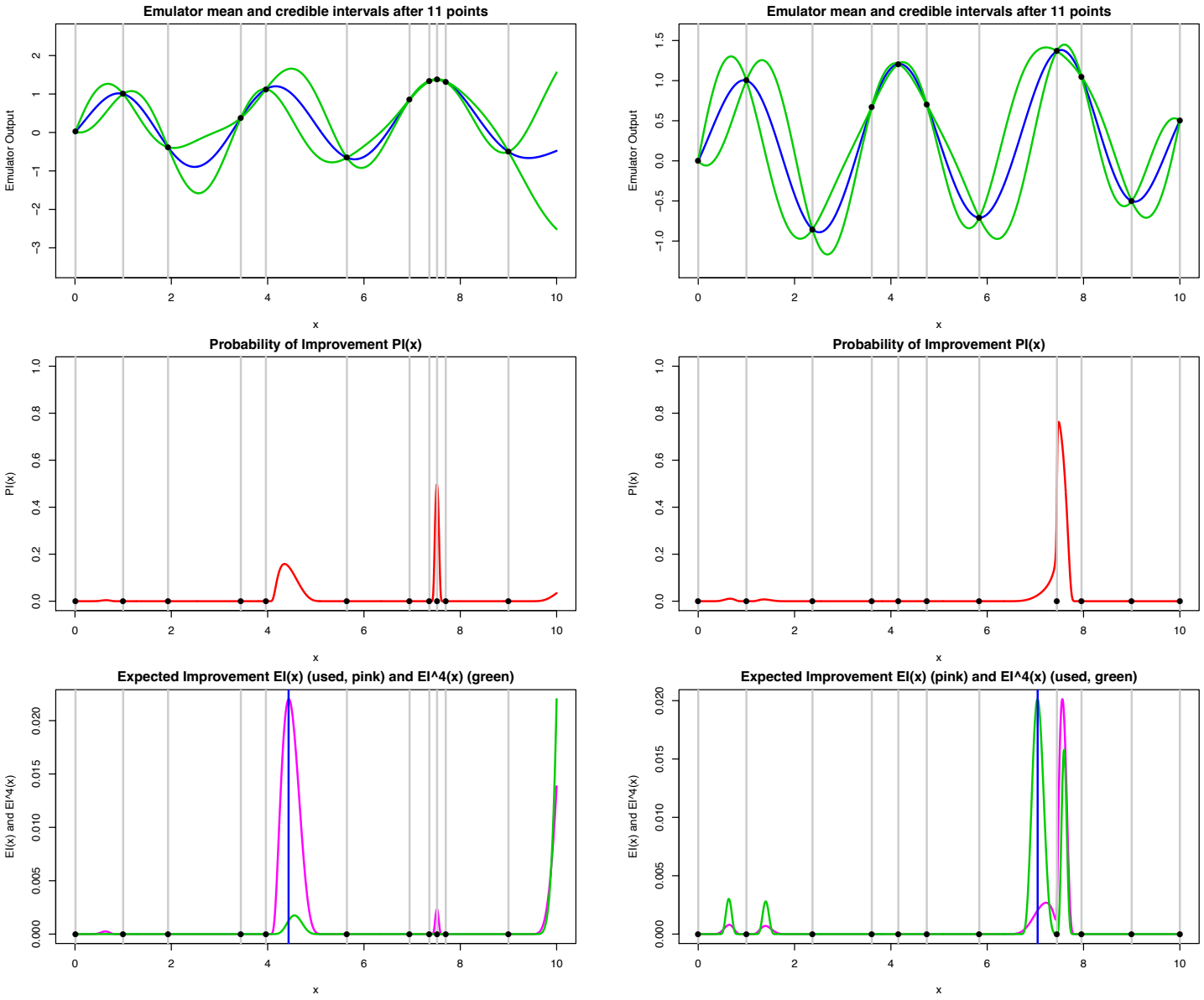


Figure 39: The 1-dimensional emulator of $f(x)$ after 11 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

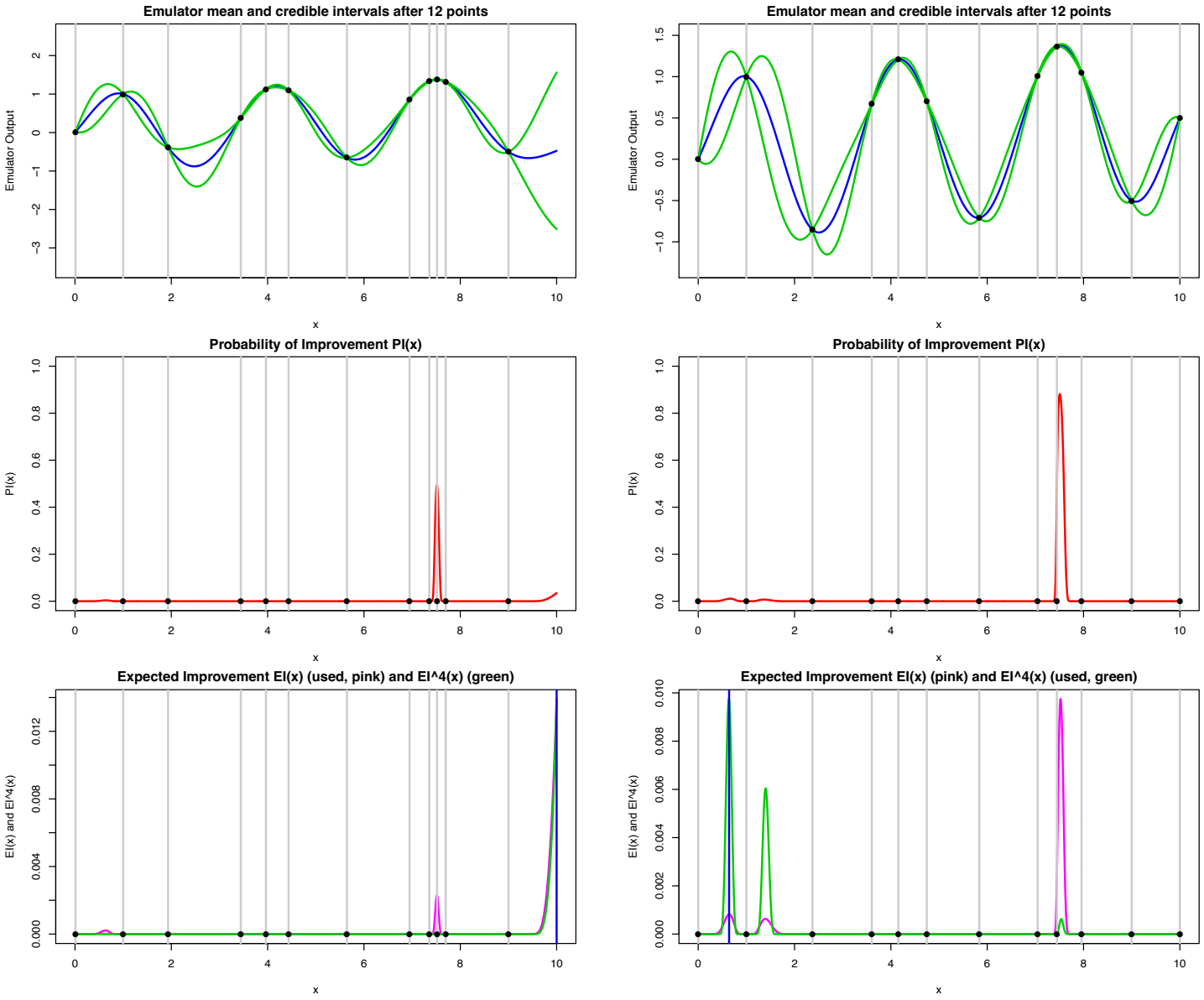


Figure 40: The 1-dimensional emulator of $f(x)$ after 12 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

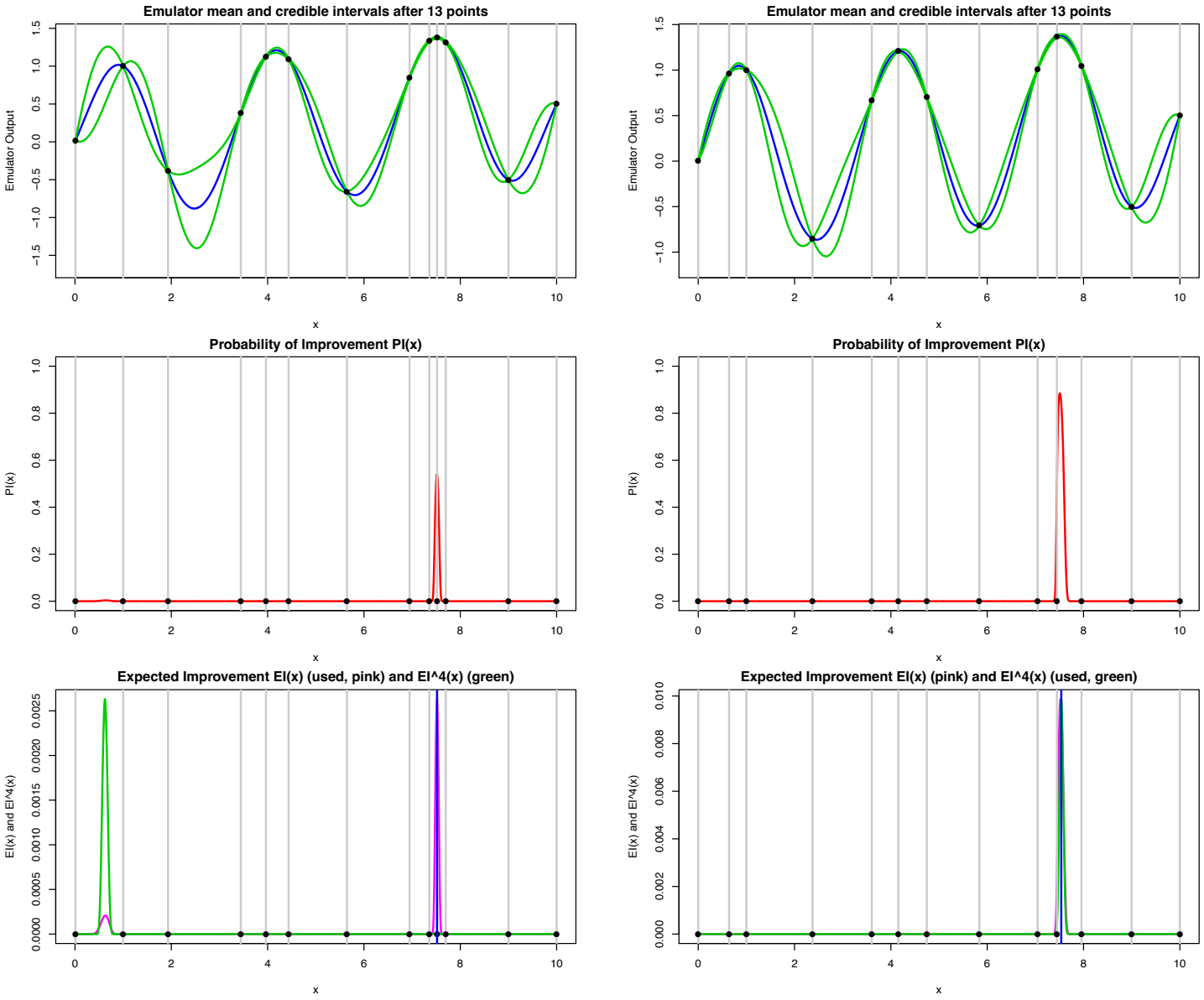


Figure 41: The 1-dimensional emulator of $f(x)$ after 13 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

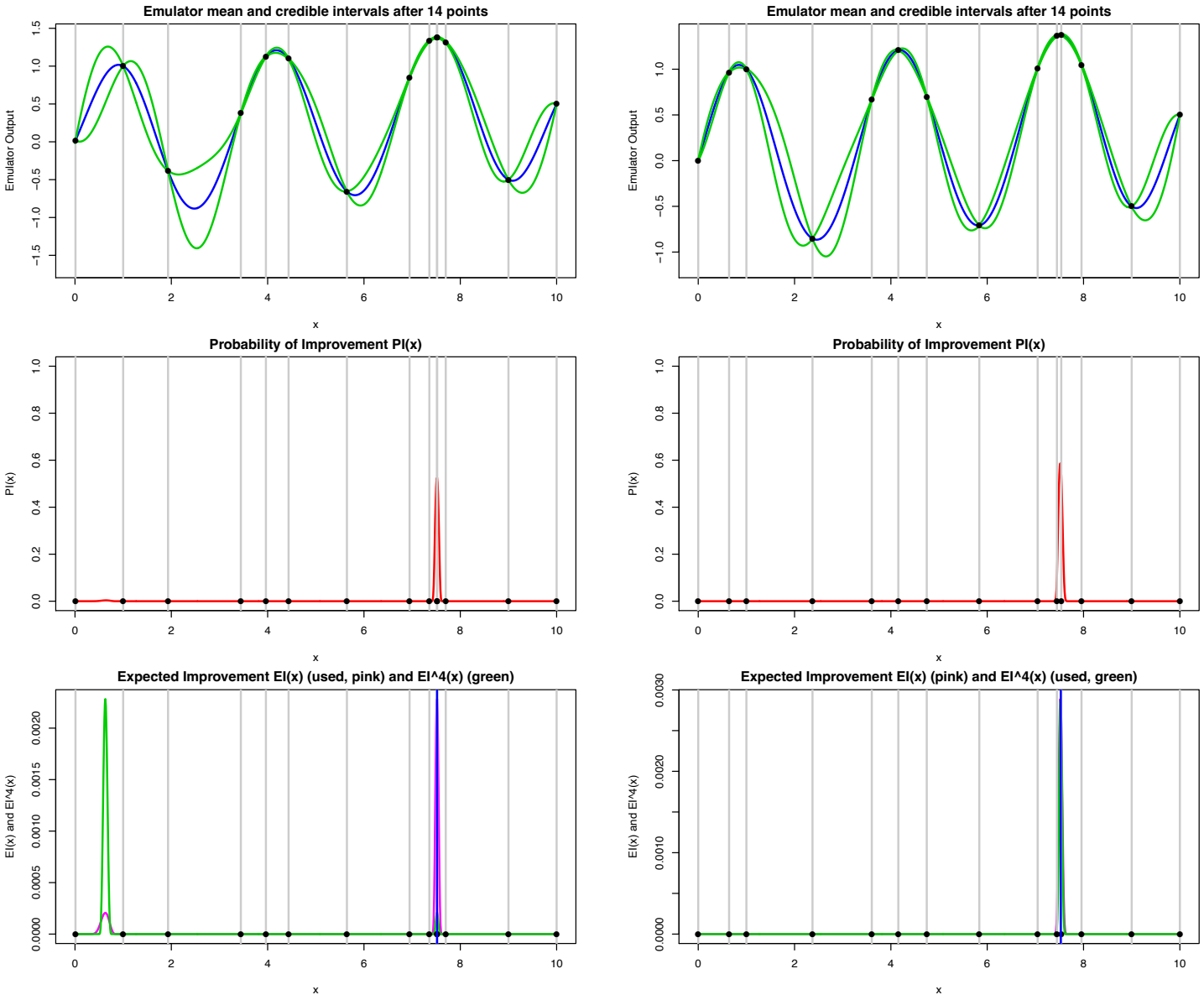


Figure 42: The 1-dimensional emulator of $f(x)$ after 14 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

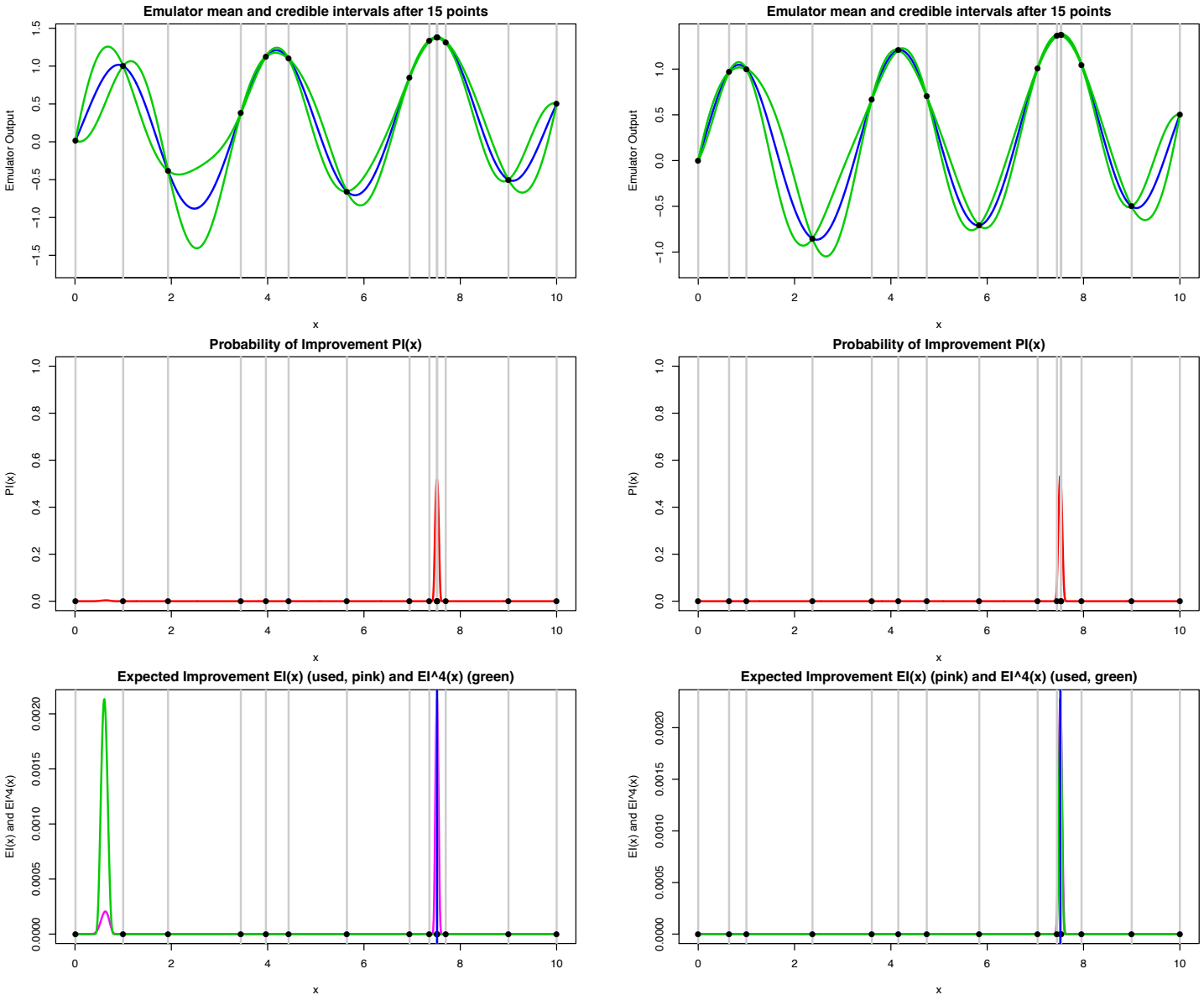


Figure 43: The 1-dimensional emulator of $f(x)$ after 15 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

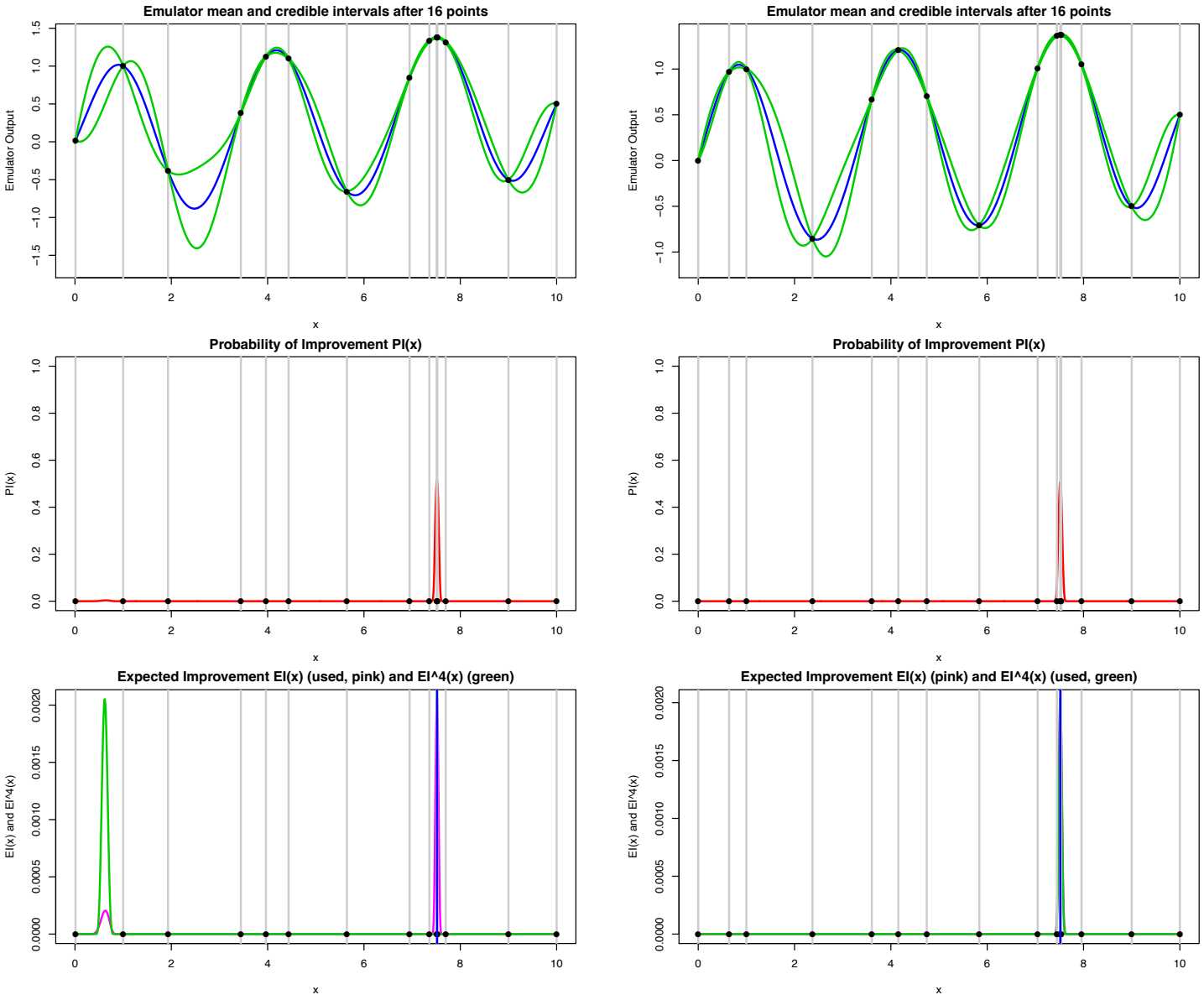


Figure 44: The 1-dimensional emulator of $f(x)$ after 16 runs. Top row, the emulator predictions: mean $\mu_D(x)$ (blue lines), credible interval $\mu_D(x) \pm 3\sigma_D(x)$ (green lines), with the evaluated runs and their locations given by the black points/grey vertical lines. Middle row: the $PI(x)$ criteria (red lines), showing substantial exploitation behaviour. Bottom row: $EI(x)$ criteria (pink lines) and the location of the next chosen input point (vertical blue line). Also shown is the alternative exploration preferring $EI_4(x)$ criteria (green lines) discussed in section 7.4. The three plots in the left column used the $EI(x)$ criteria to choose the next point. The three plots in the right column used the $EI_4(x)$ criteria to choose the next point.

Example: Bayesian Optimisation in 2D

- We now examine the behaviour of both the $PI(x)$ and $EI(x)$ acquisition functions on a 2-dimensional case.
- In section 7.4 we will again return to this example with a more exploration preferring criteria.
- We wish to emulate and optimise the 2-dimensional function

$$f(x) \quad = \quad \cos(4\pi x_1 - \pi) + \sin(2\pi x_2) \tag{152}$$

  over the region $\mathcal{X} = \{x \in \mathbb{R}^2 : 0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$, which was again deliberately chosen to have two modes.
- We again construct a simple zero mean Gaussian process emulator, with the correlation length $\theta = 0.3$, prior variance $\sigma_u^2 = 1^2$, zero regression terms so $\beta_{ij} = 0$ and a small nugget for numerical stability $\sigma_w^2 = 10^{-6}\sigma_u^2$.
- This time we assume three runs have already been performed at locations $x^{(1)} = (0.3, 0.8), x^{(2)} = (0.7, 0.8)$ and $x^{(3)} = (0.53, 0.3)$, with the latter point chosen slightly off centre to break the reflection symmetry (which would present even more serious problems for the $PI(x)$ criteria).
- Figure 45 shows the form of the emulator of $f(x)$ and various acquisition functions, once the evaluation of the above three runs have been performed.
- The top left panel shows the true function $f(x)$ over the 2D input space $\mathcal{X}$ as a contour plot, with the white points showing the locations of the evaluated set of runs. The two maxima are shown as black crosses which we will attempt to locate. The top right panel shows the emulator expectation or mean $\mu_D(x)$, and the middle left panel shows the emulator standard deviation $\sigma_D(x)$. The middle right panel gives the $PI(x)$ criteria, the bottom left the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which will be discussed and used in section 7.4).
- We note as expected, the emulator standard deviation is low (light blue) around the evaluated points and high (pink) elsewhere, and also that the emulator mean is not capturing the function $f(x)$ well after only three points.

Using the $PI(x)$ Criteria

- We first use the $PI(x)$ criteria to optimise the function. Figures 46 to 61 show the evolution of the same six plots, as we sequentially add new input points based on the $PI(x)$ acquisition function (given in the middle right panel). In each figure, the latest input point is shown as the pink point, with all previous run locations given as white points.

- The $PI(x)$ criteria performs badly: it only advocates small, incremental improvements, which results in a slow convergence to only one of the two maxima locations, and displays no exploration at all. It does however arrive at the maxima after 21 points (see figure 58), but after this it persistently places the next nine points at the same maxima, ignoring the possibility of the existence of the second maxima (see figures 59 and 61), however this is related to the inclusion of a nugget, which implies there is always some uncertainty even close to known runs.

- Note that to generate these contour plots and to perform the maximisation over the inputs, we have used a grid x_P of 61×61 points, which leads to a slightly discretised path when using $PI(x)$, but which should not effect the general results.

Using the $EI(x)$ Criteria

- We now use the Expected Improvement $EI(x)$ criteria to optimise the function. Figures 62 to 78 show the evolution of the same six plots, as we sequentially add new input points based on the $EI(x)$ acquisition function, which is shown in the bottom left panel.

- Initially, the $EI(x)$ criteria chooses 9 additional points in exploratory locations, mainly where the emulator variance is high along boundaries and in some corners (figures 62 to 71). Then it clearly discovers the right of centre peak, and proceeds to choose points to locate the maxima of this mode (figures 72 to 75). However, it then gets stuck on this mode (again, this is related to the use of a nugget), and wastes the last 9 points on or very close to this mode (figures 76 to 78). It does not find the second maxima, or even the general location of it at all. In repeated runs of this experiment, with different starting points, the $EI(x)$ criteria sometimes will locate the second maxima, but not often.

- Finding multiple maxima is of course a hard problem, and we should not be too surprised that $EI(x)$ does not always achieve this. In many cases it will succeed, given enough runs, and depending on the particulars of the function $f(x)$ and the details of the emulator specification.

- This shows that the balance between exploration and exploitation is even more critical in higher dimensions, and that we may wish to improve the form of the $EI(x)$ criteria, depending on what our preferences are, as we now go on to describe.

- To facilitate this discussion, first we will recast these acquisition functions and optimisation strategies in terms of Decision Theory.

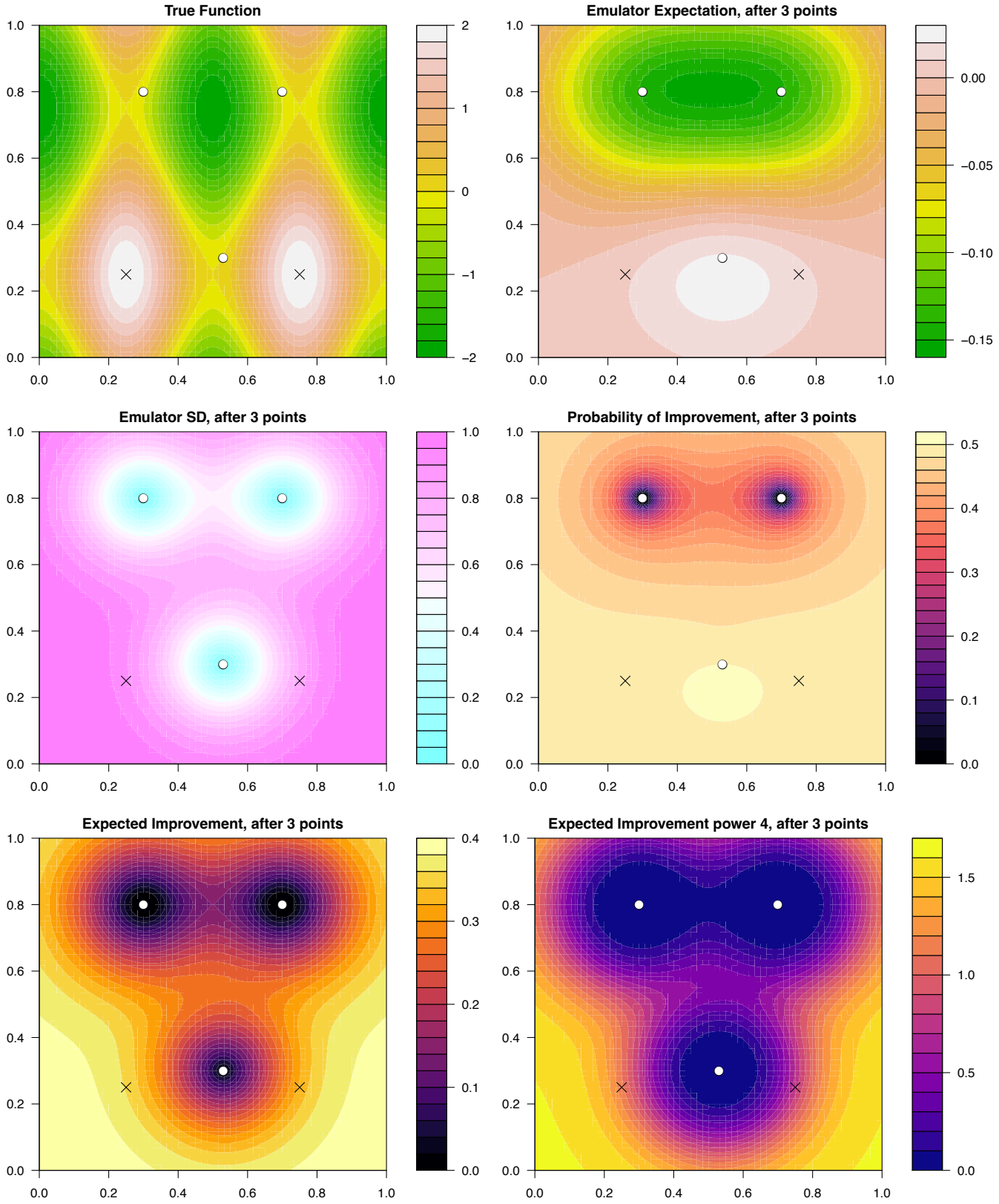


Figure 45: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

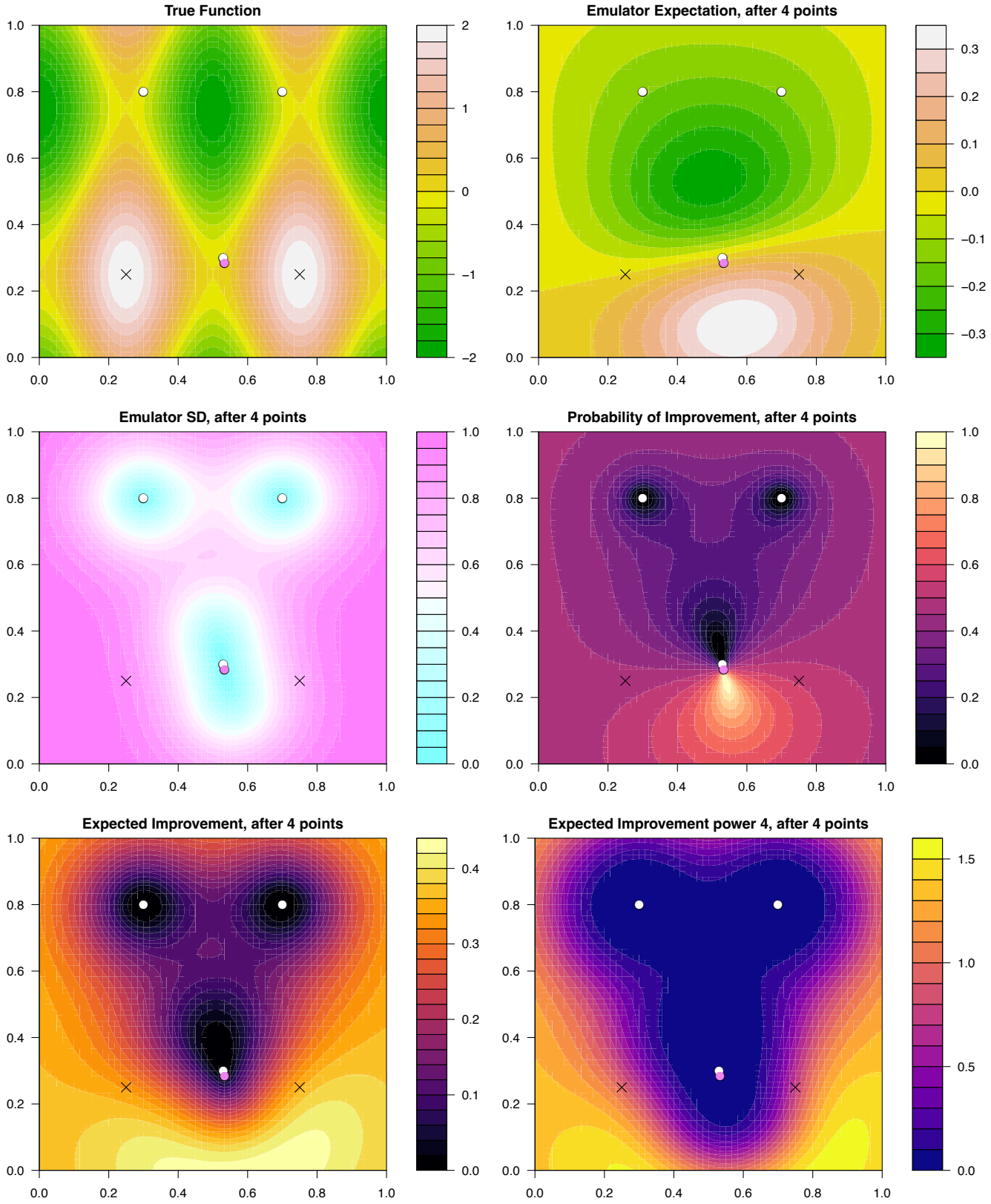


Figure 46: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

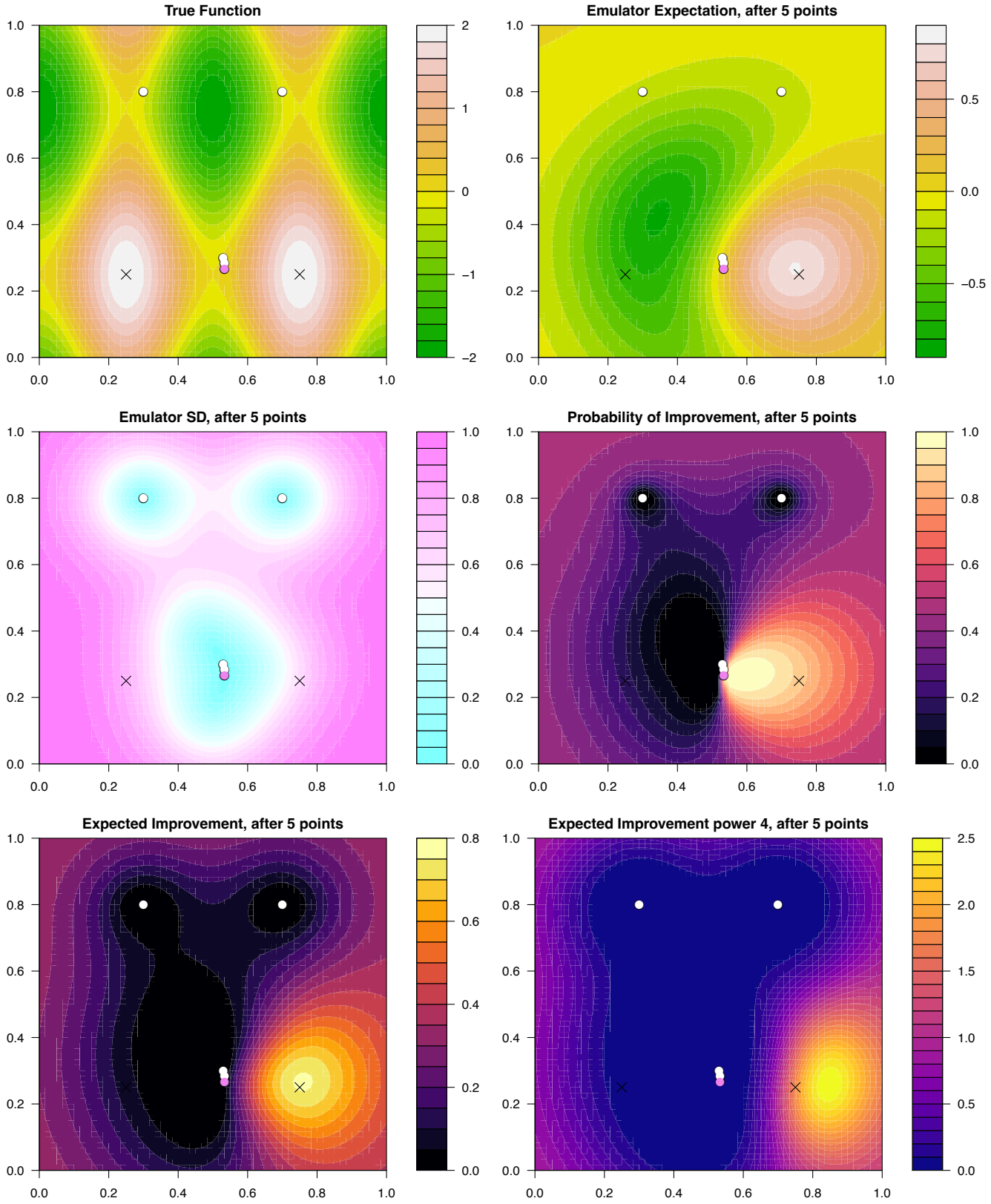


Figure 47: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

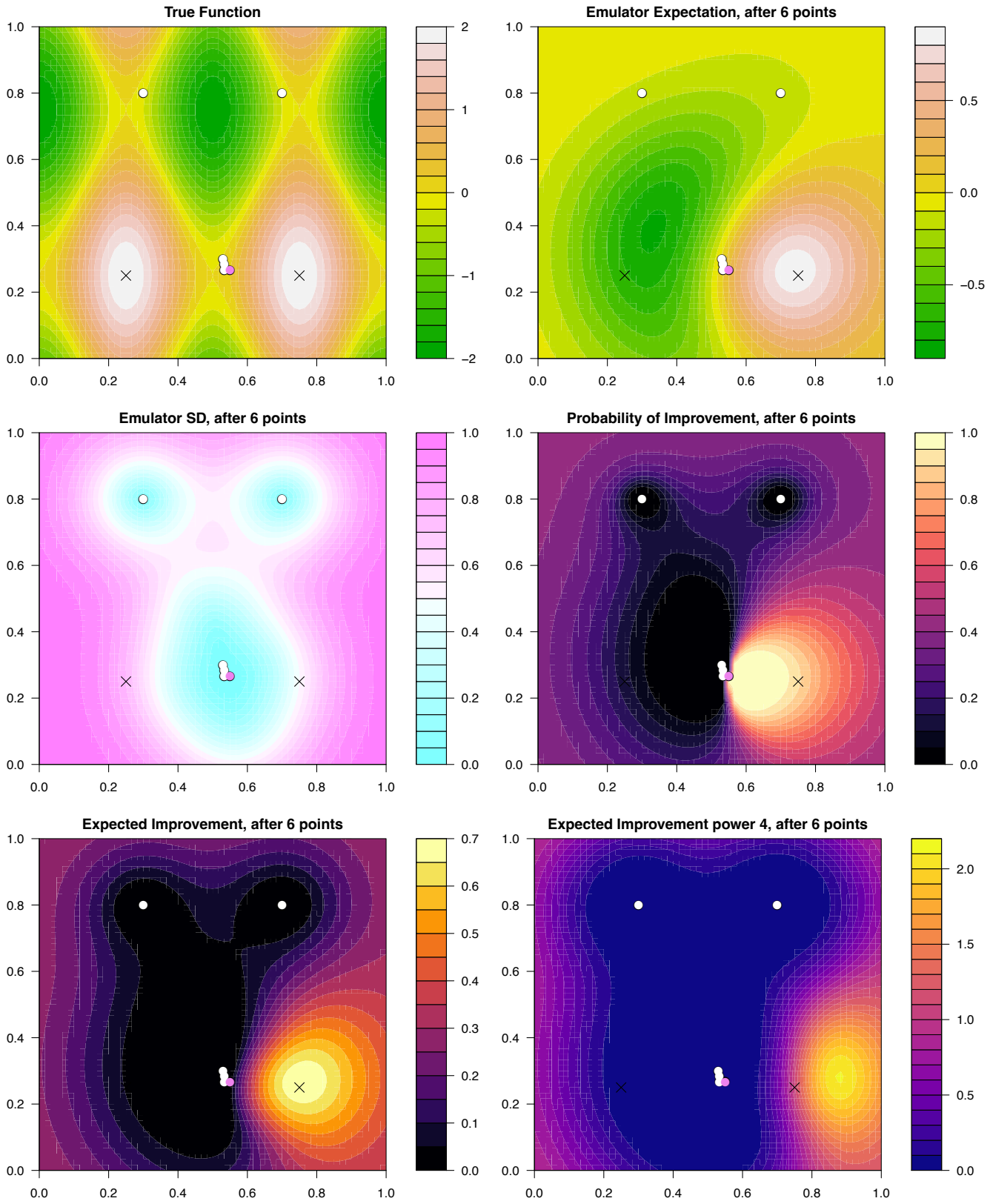


Figure 48: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

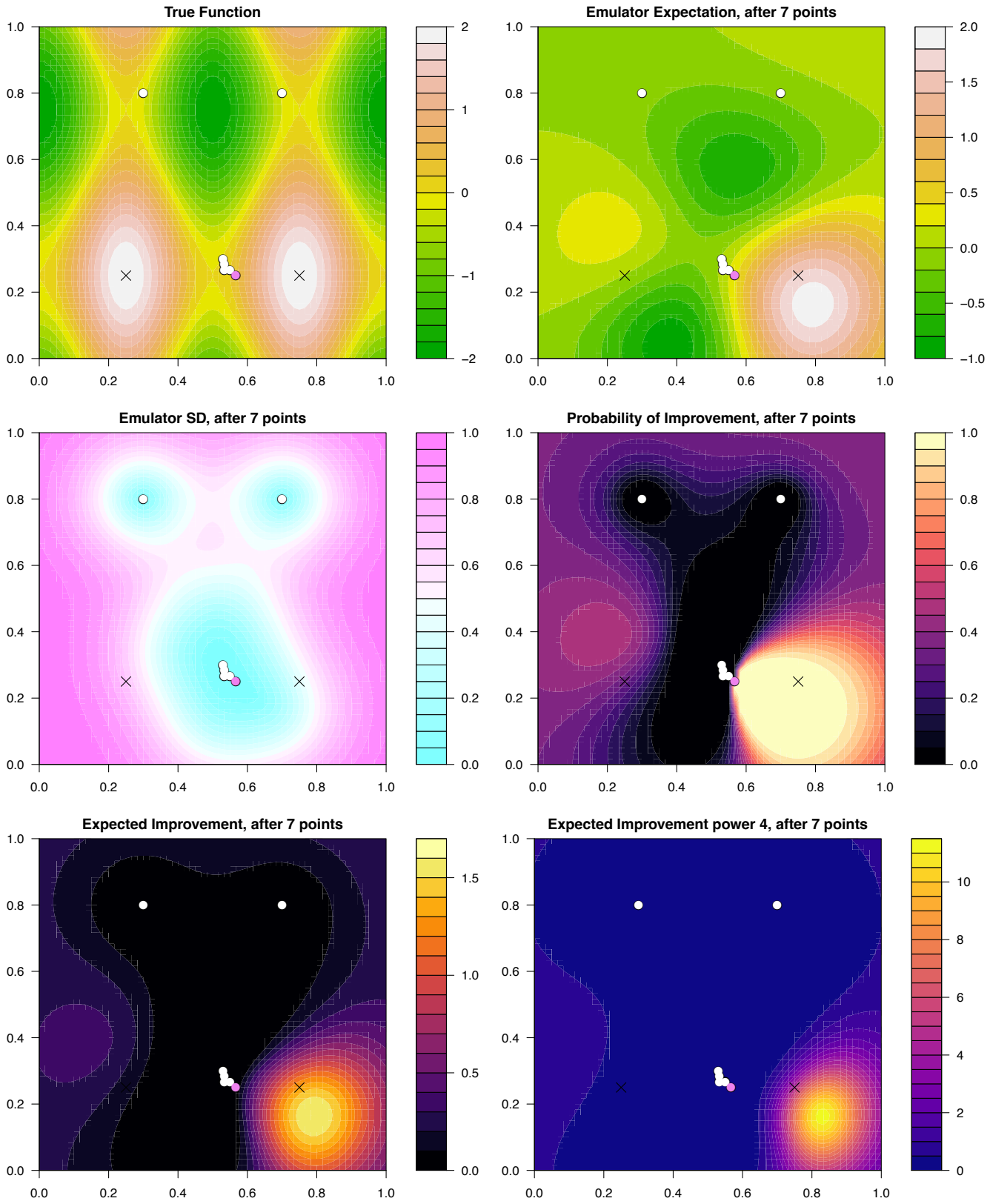


Figure 49: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

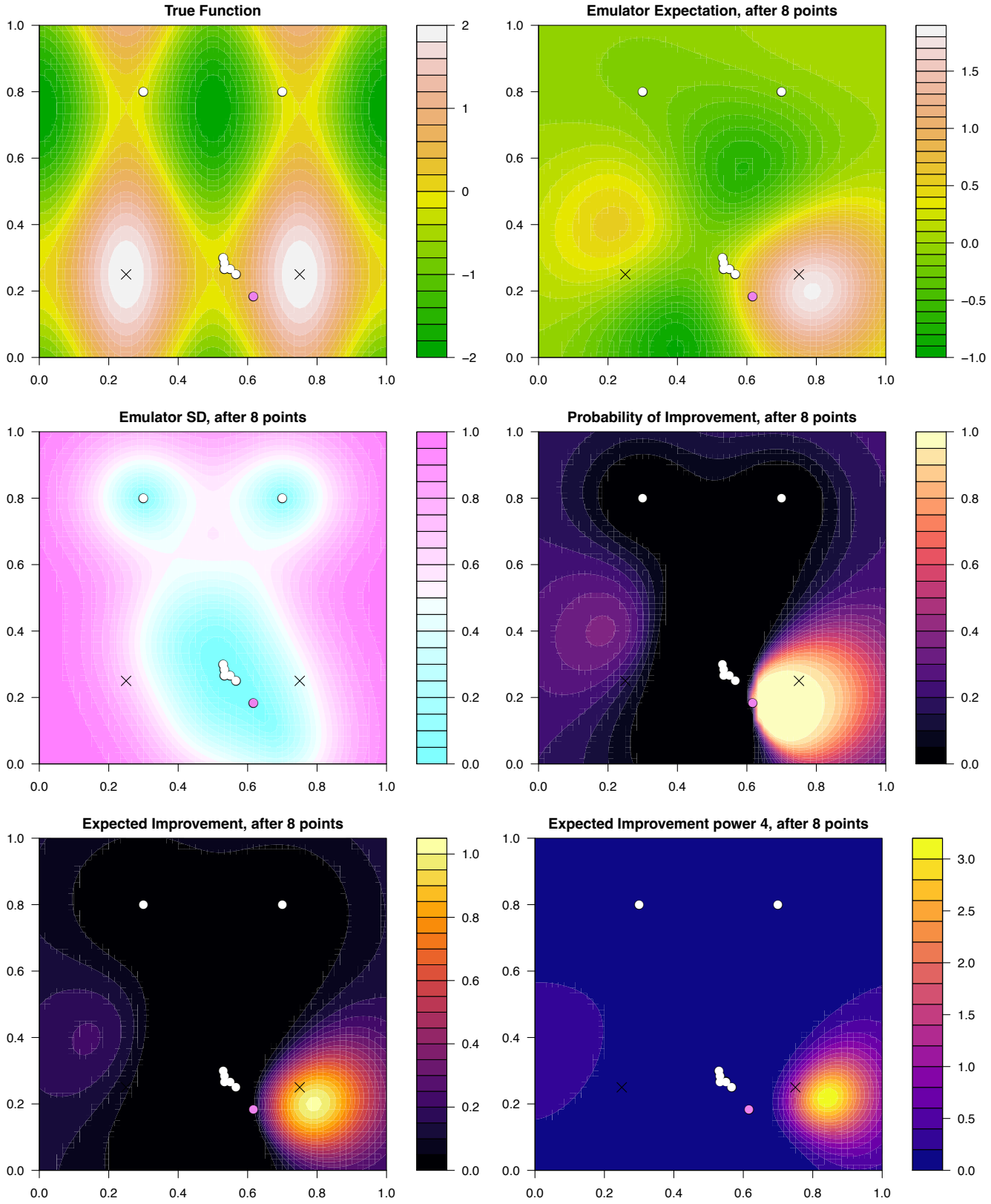


Figure 50: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

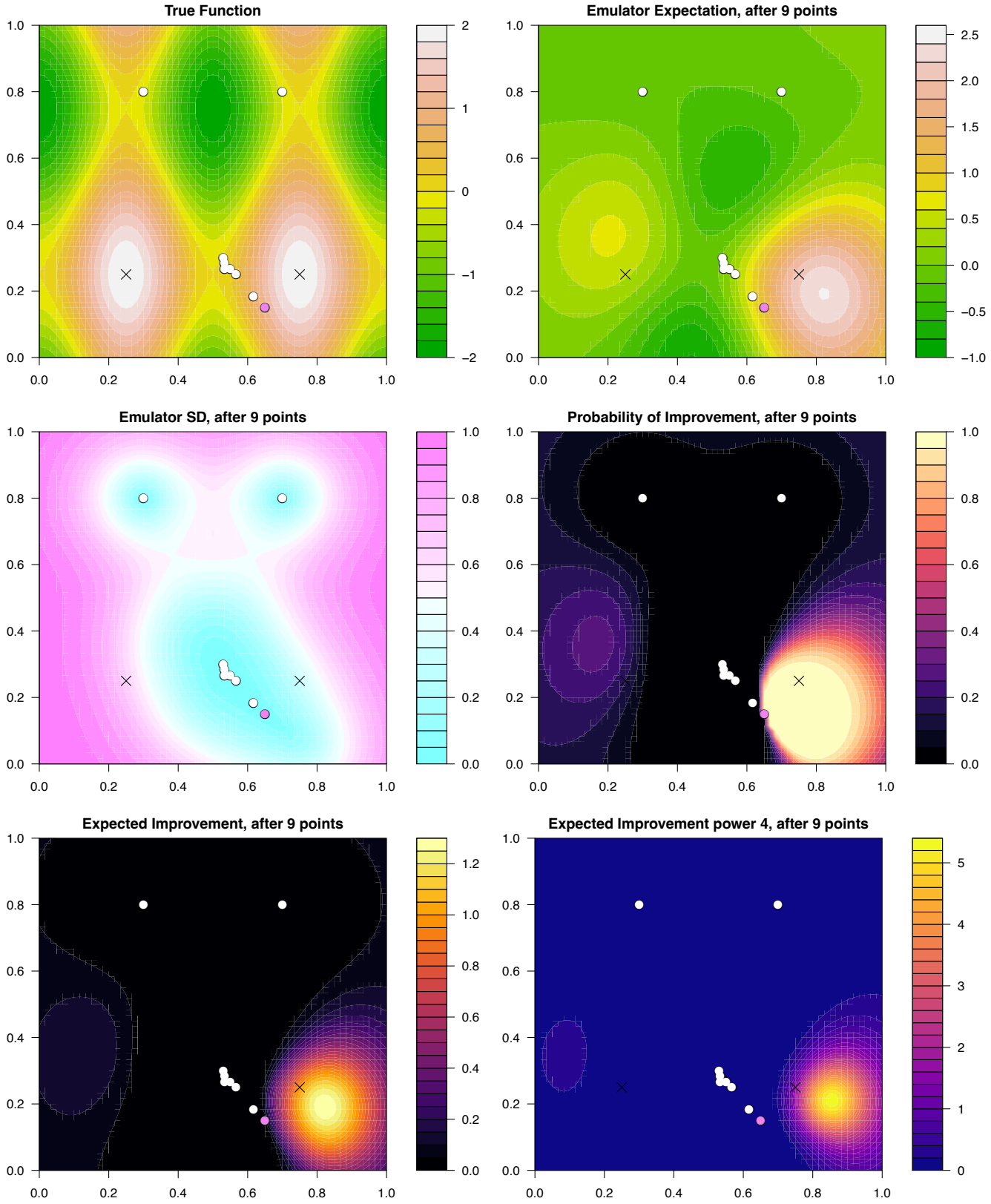


Figure 51: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

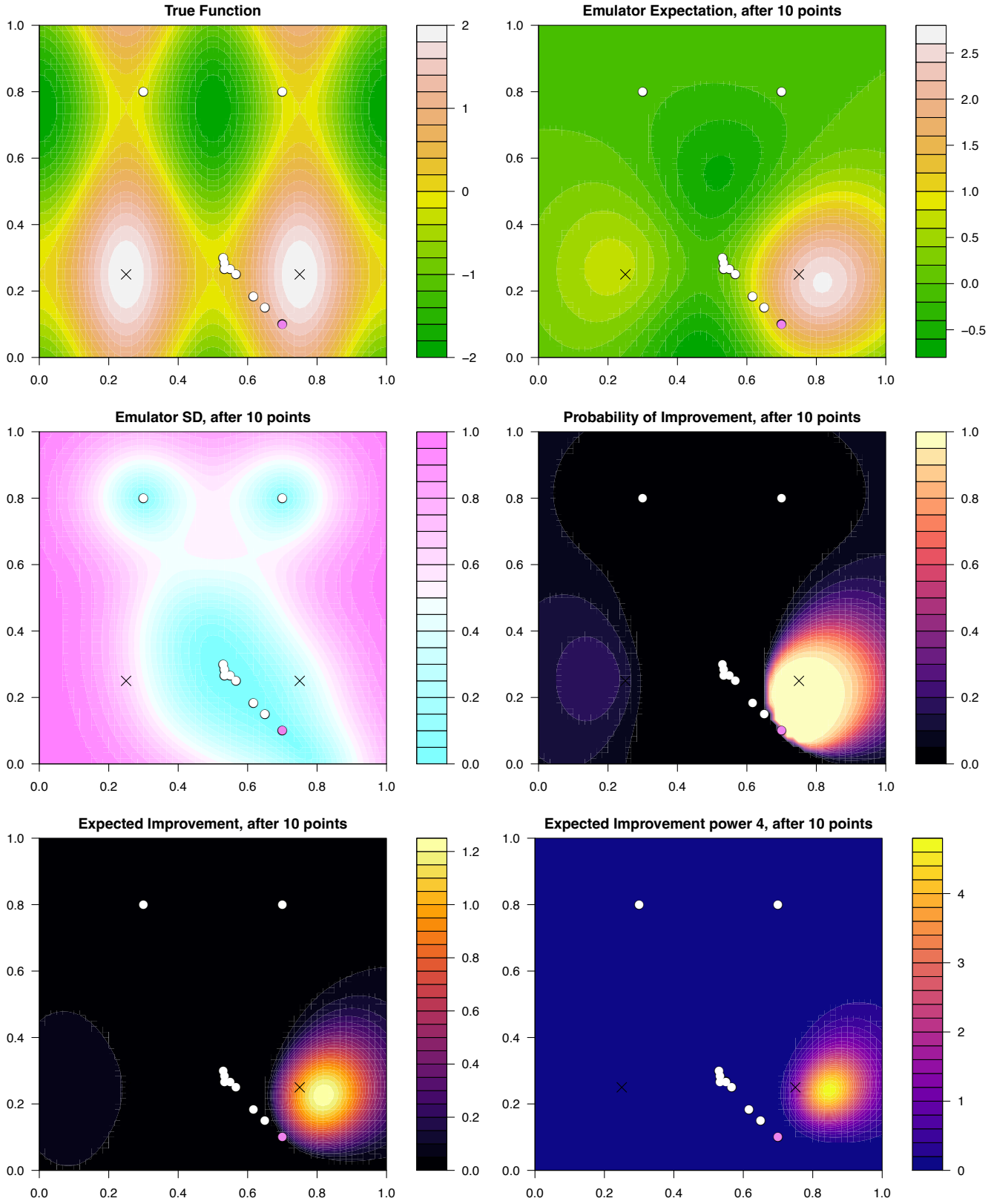


Figure 52: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

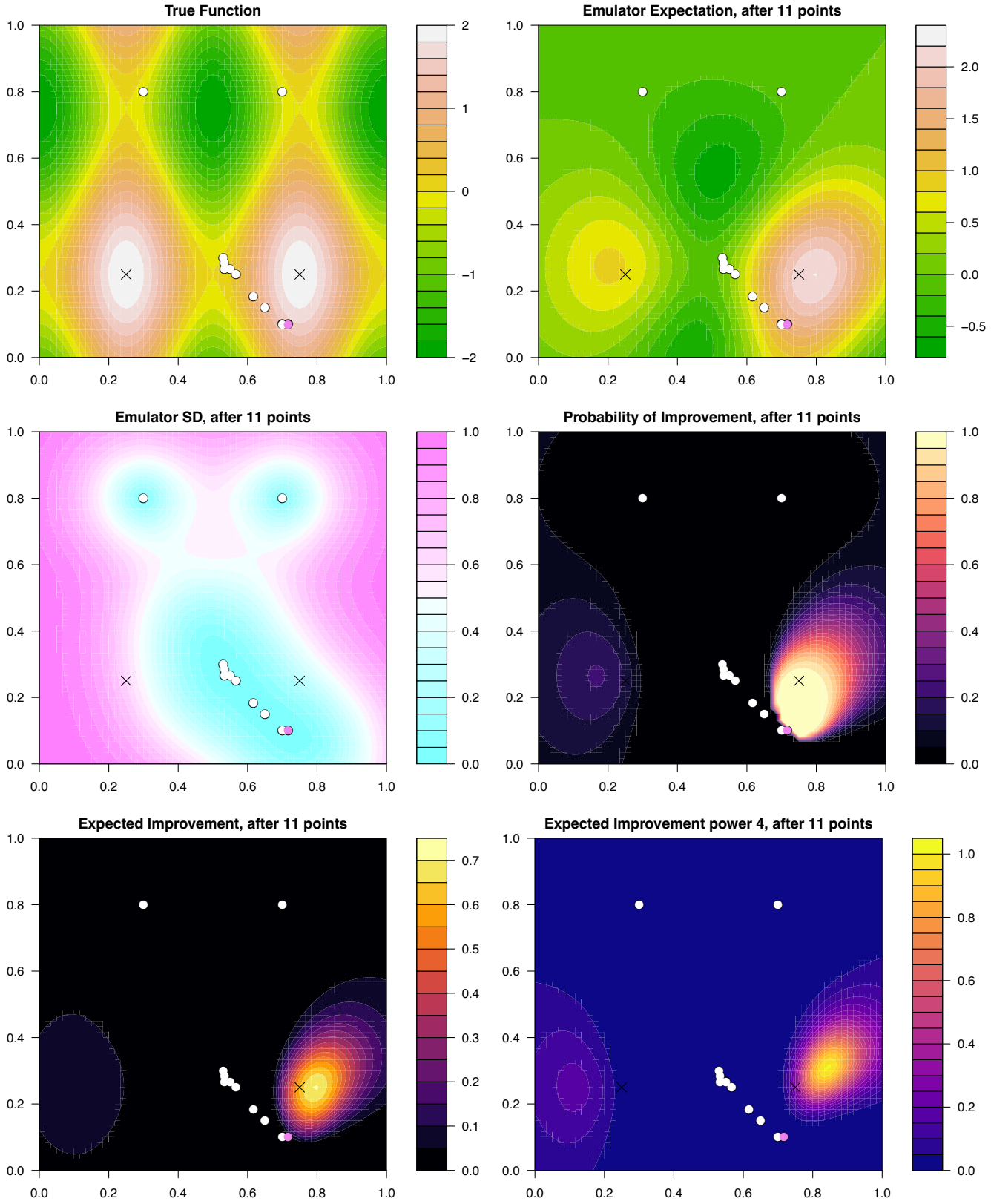


Figure 53: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

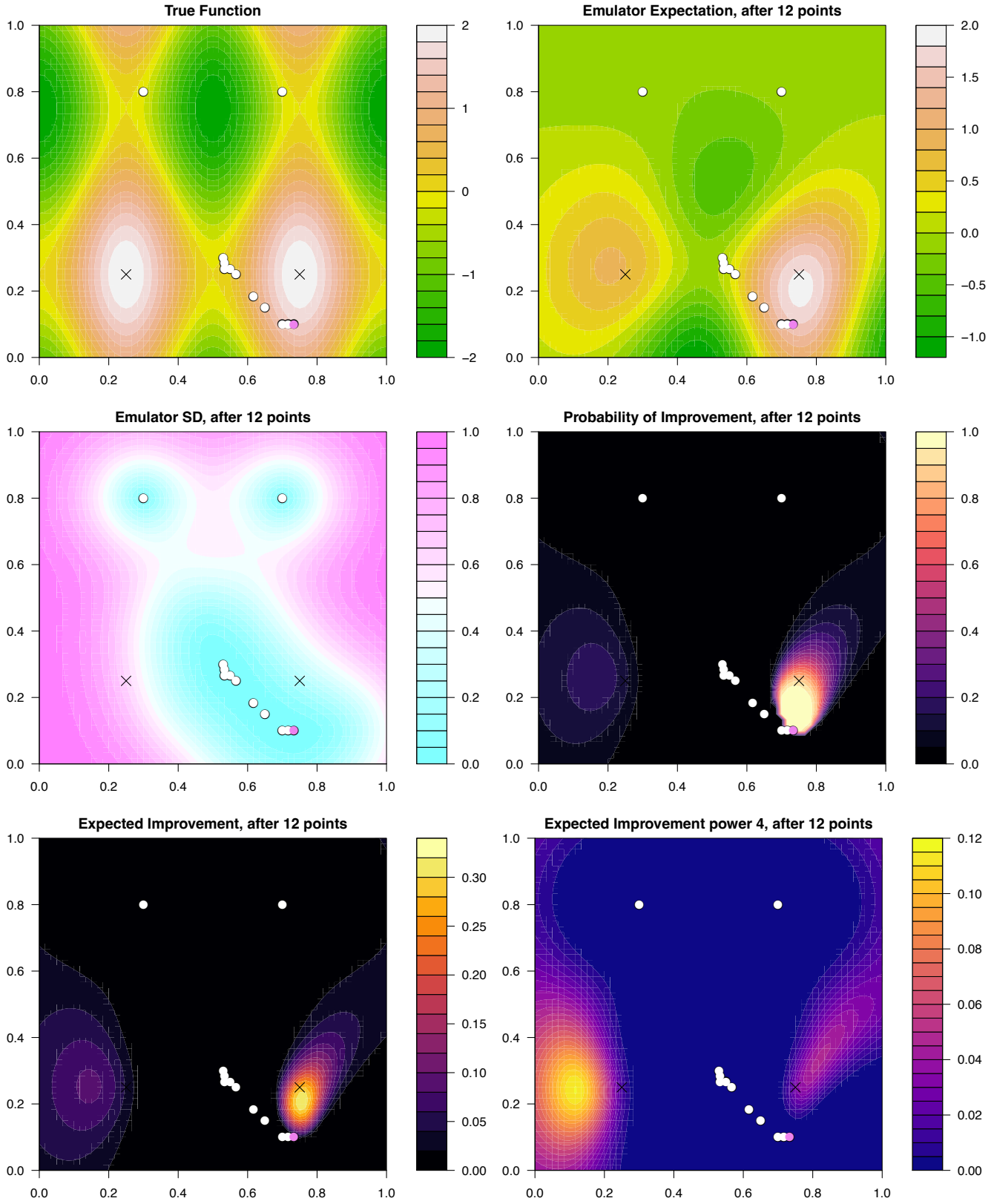


Figure 54: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

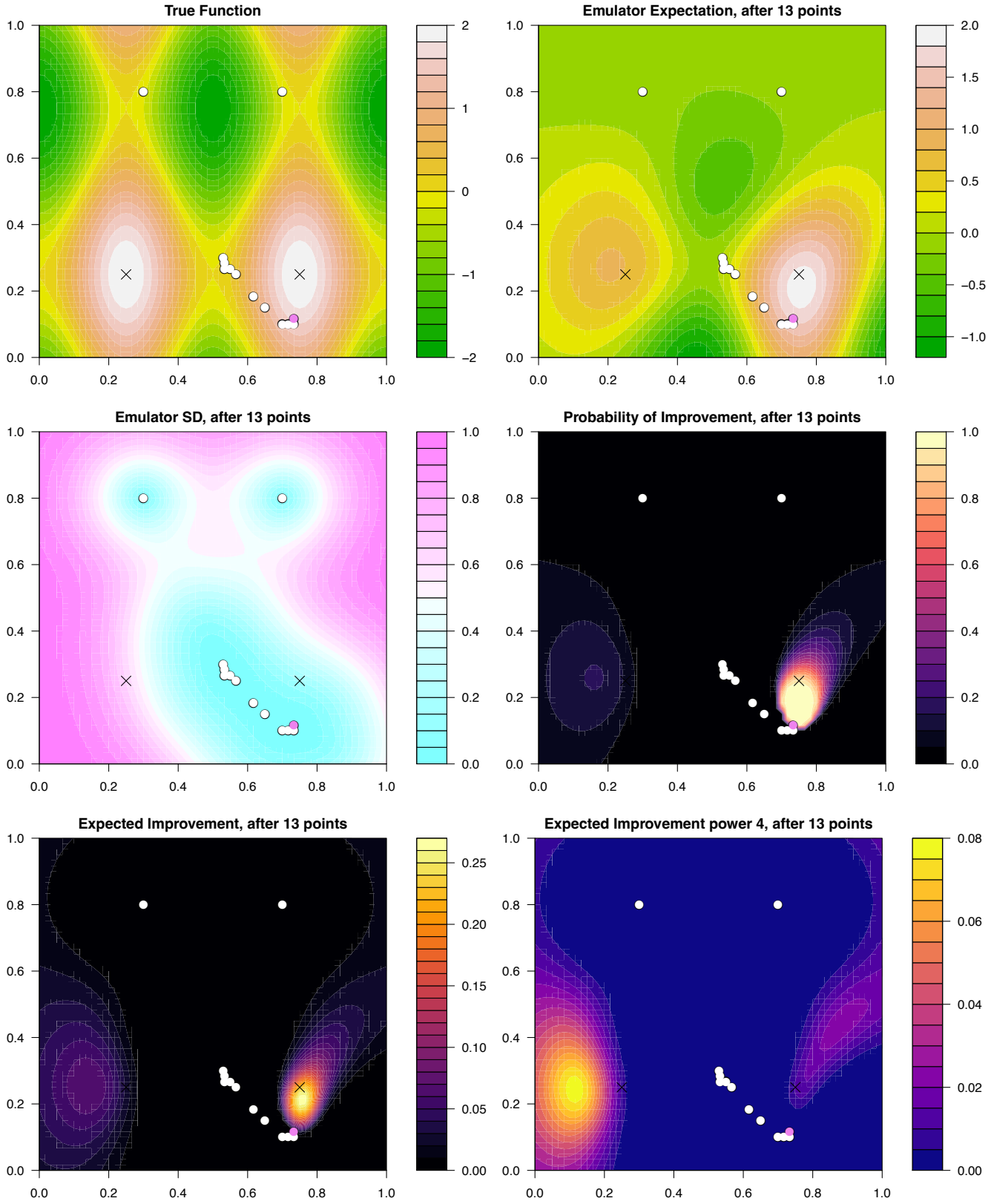


Figure 55: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

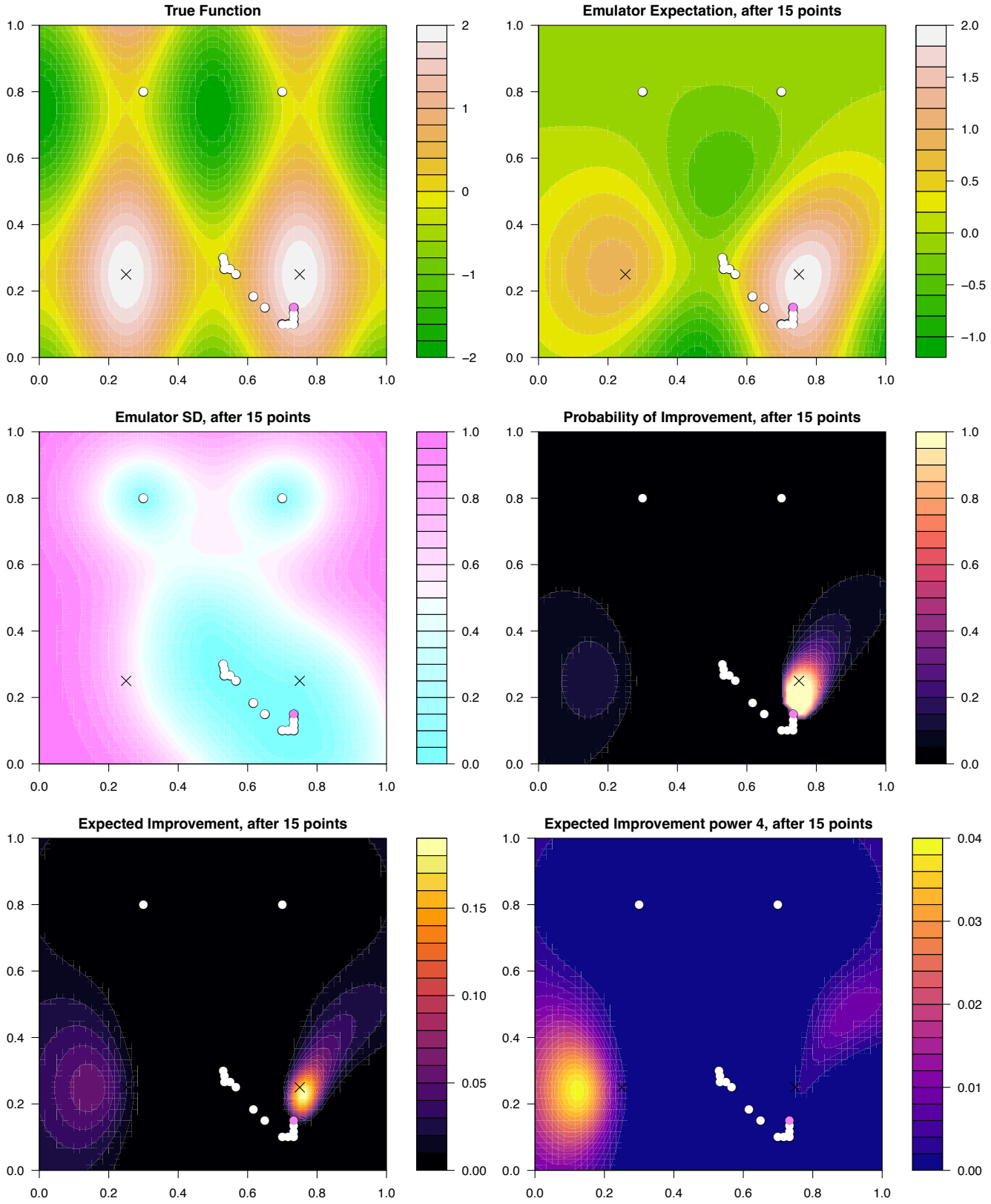


Figure 56: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

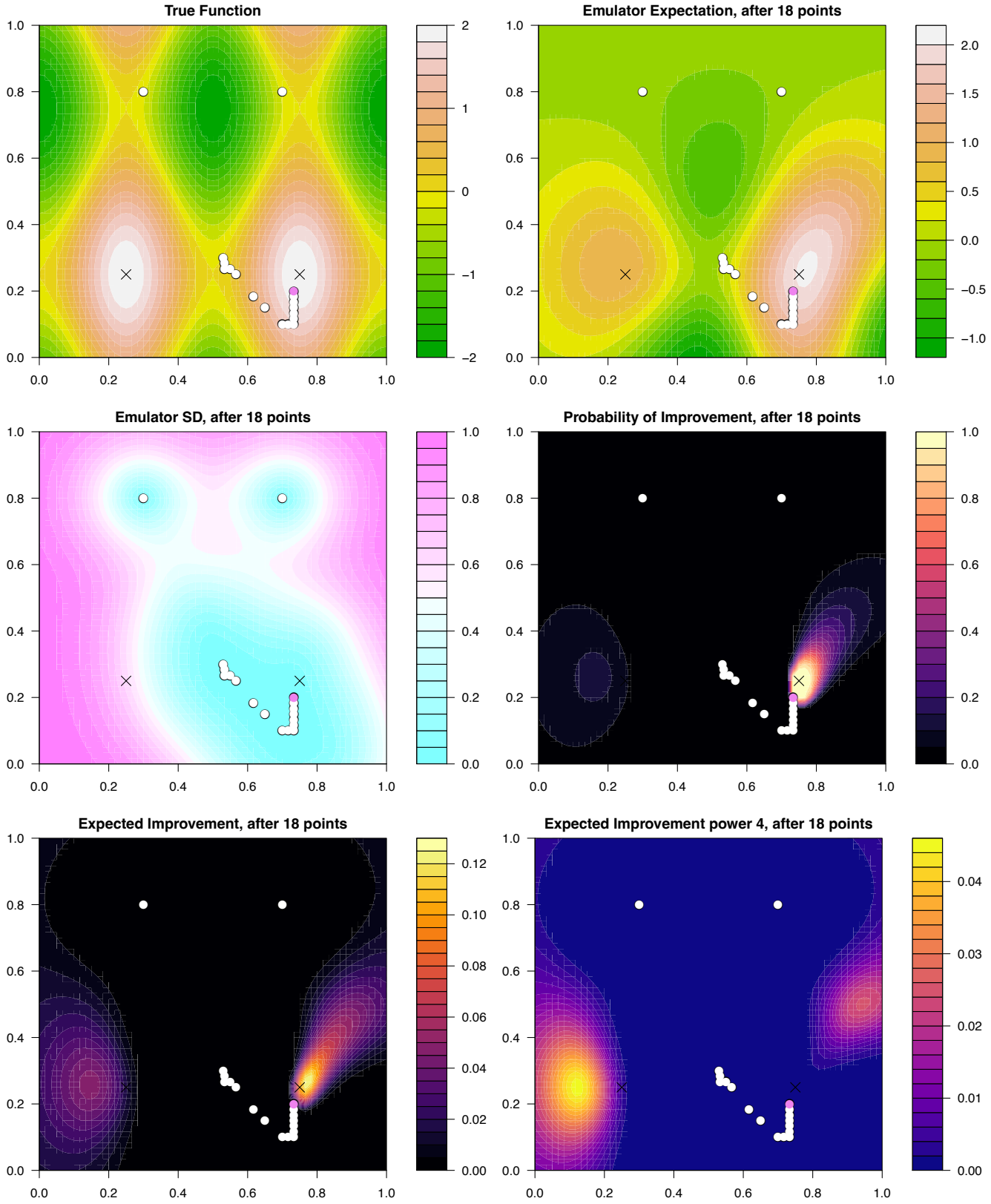


Figure 57: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

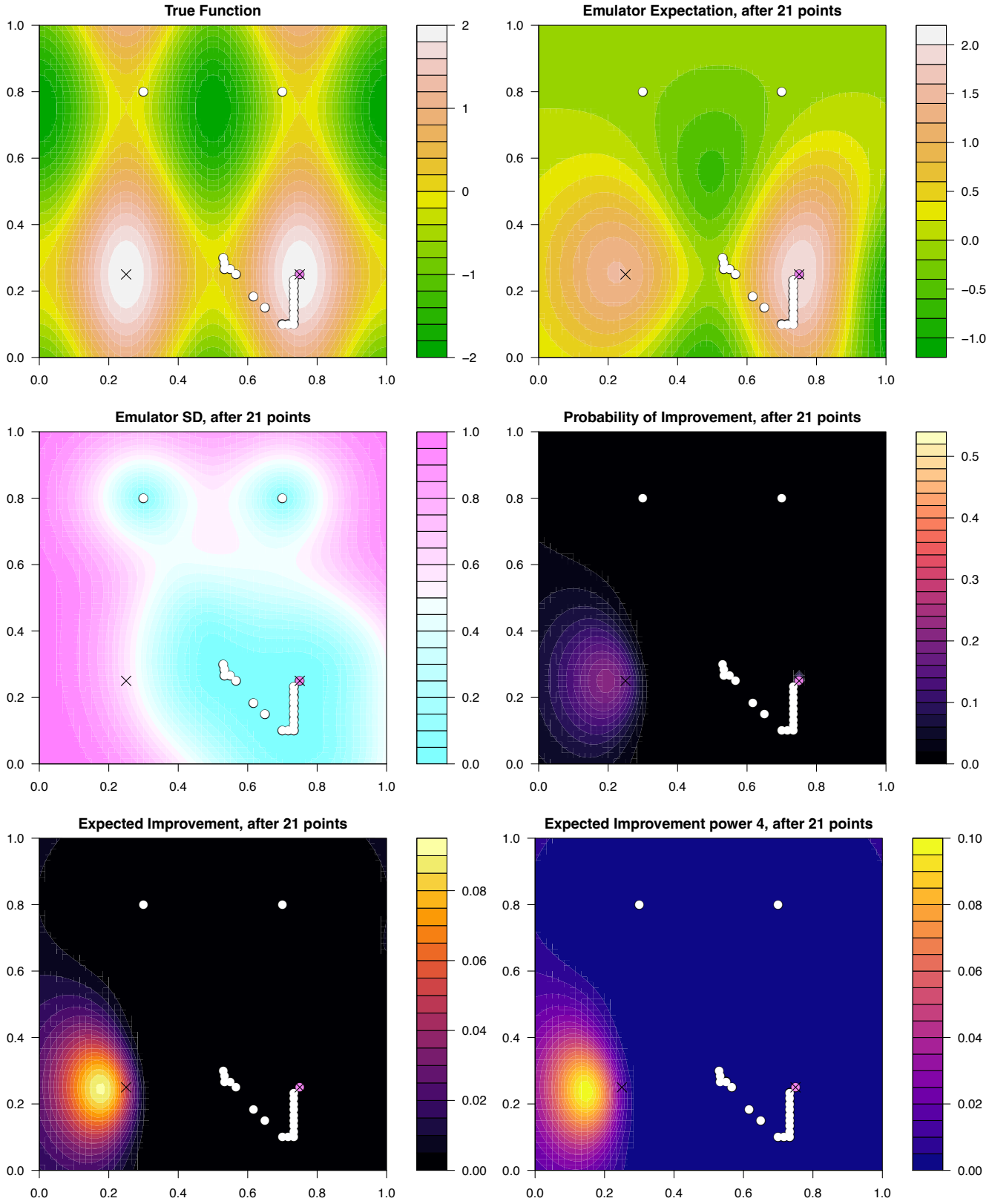


Figure 58: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

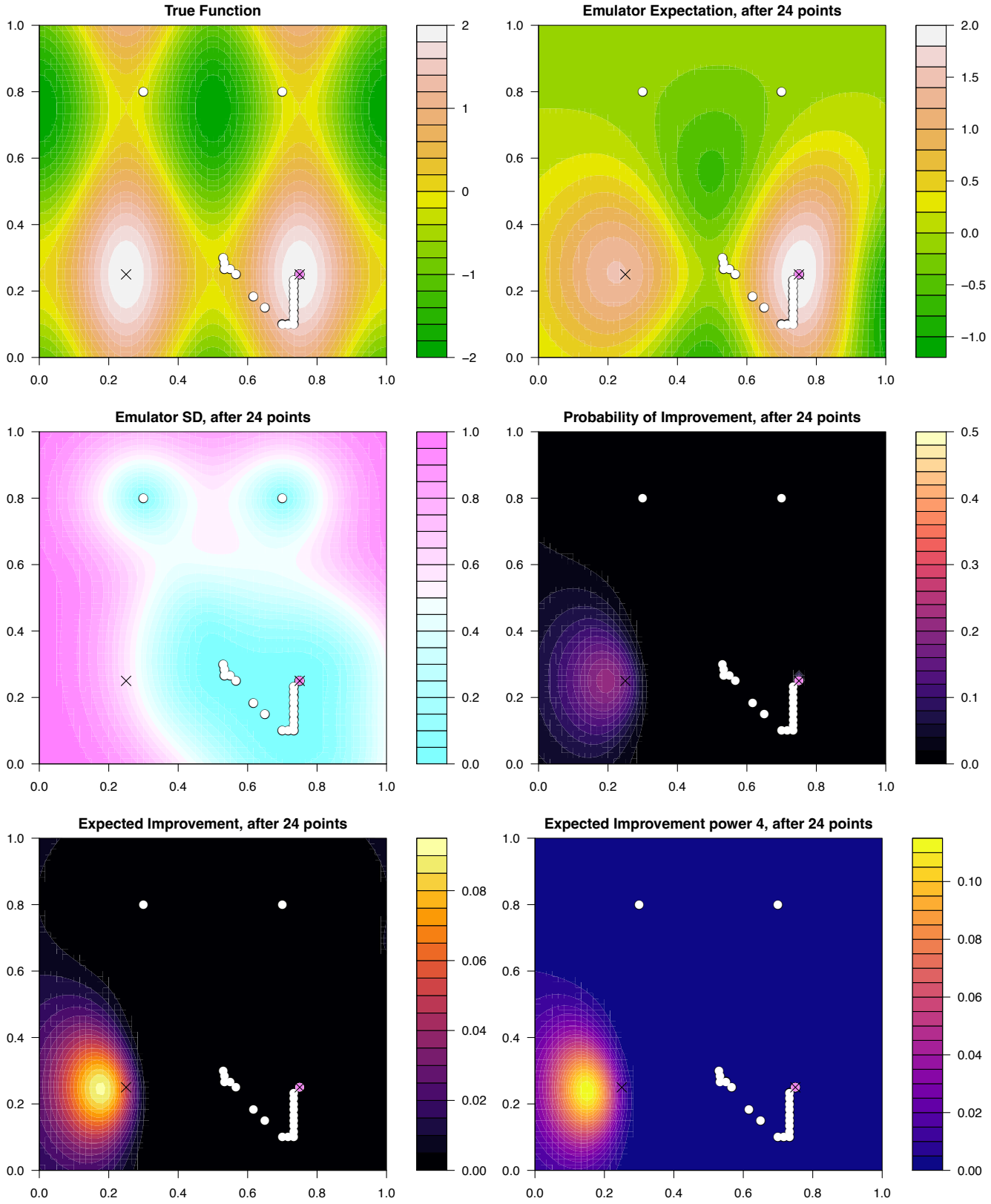


Figure 59: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

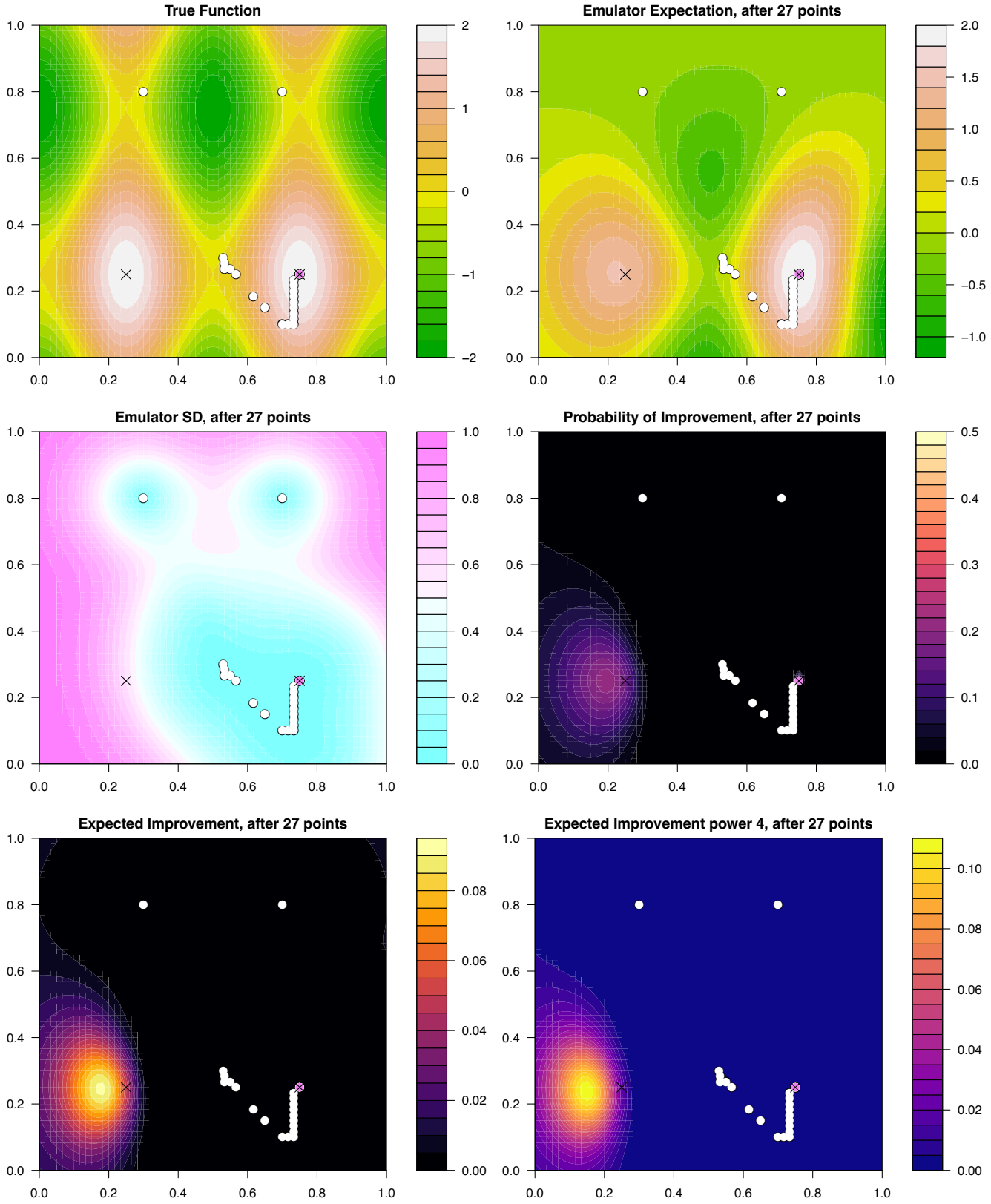


Figure 60: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

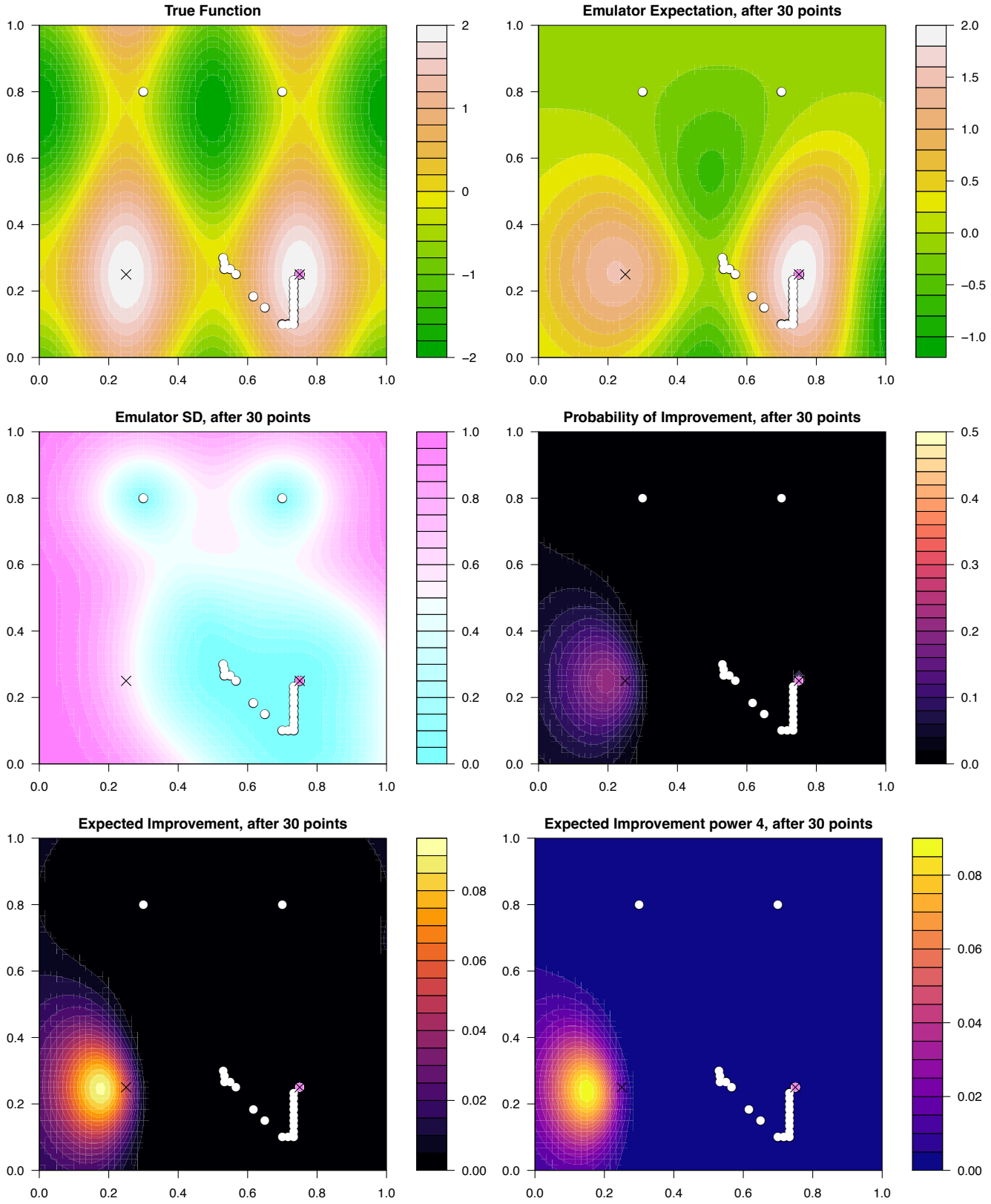


Figure 61: **Optimising using the $PI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

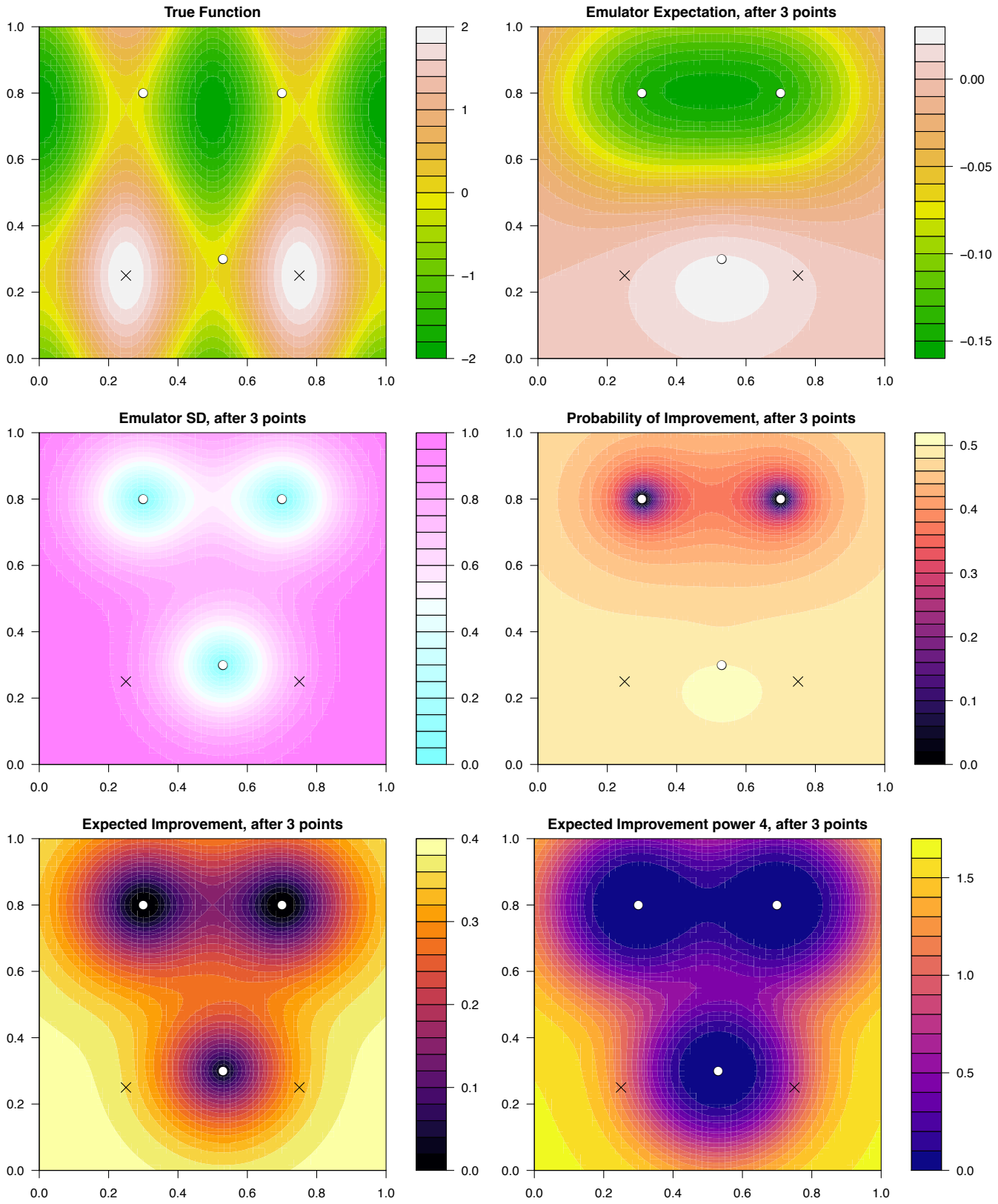


Figure 62: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

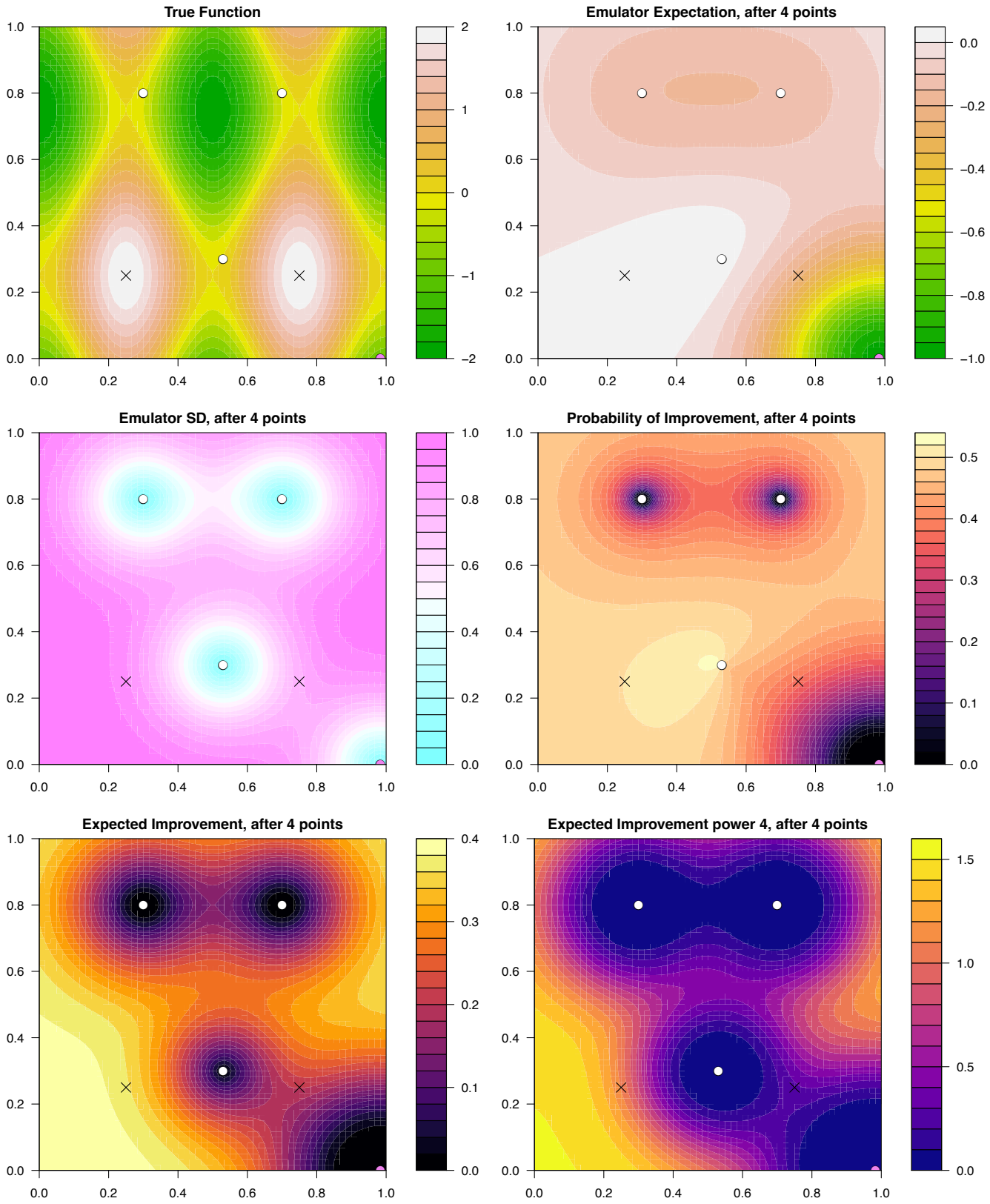


Figure 63: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

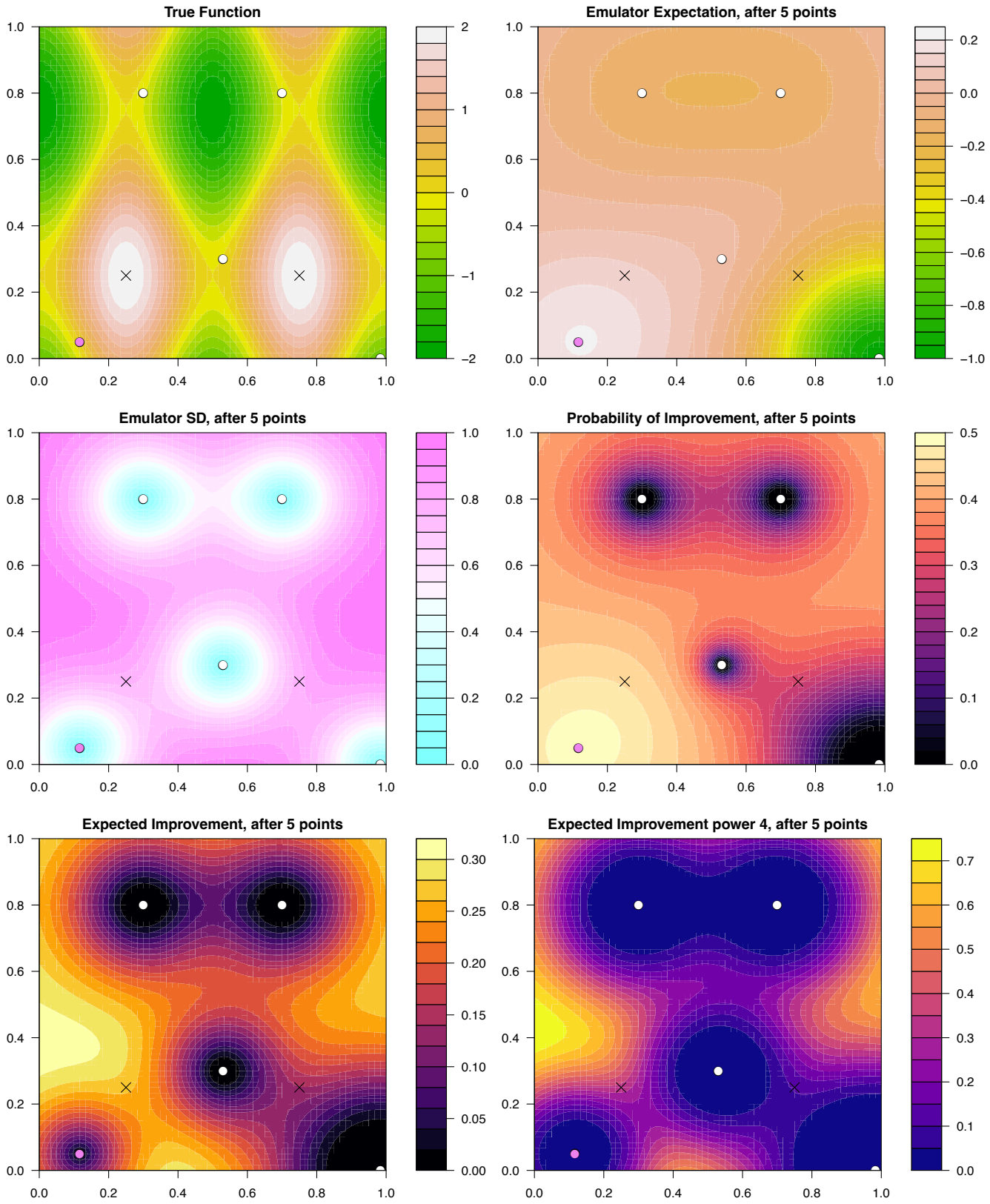


Figure 64: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

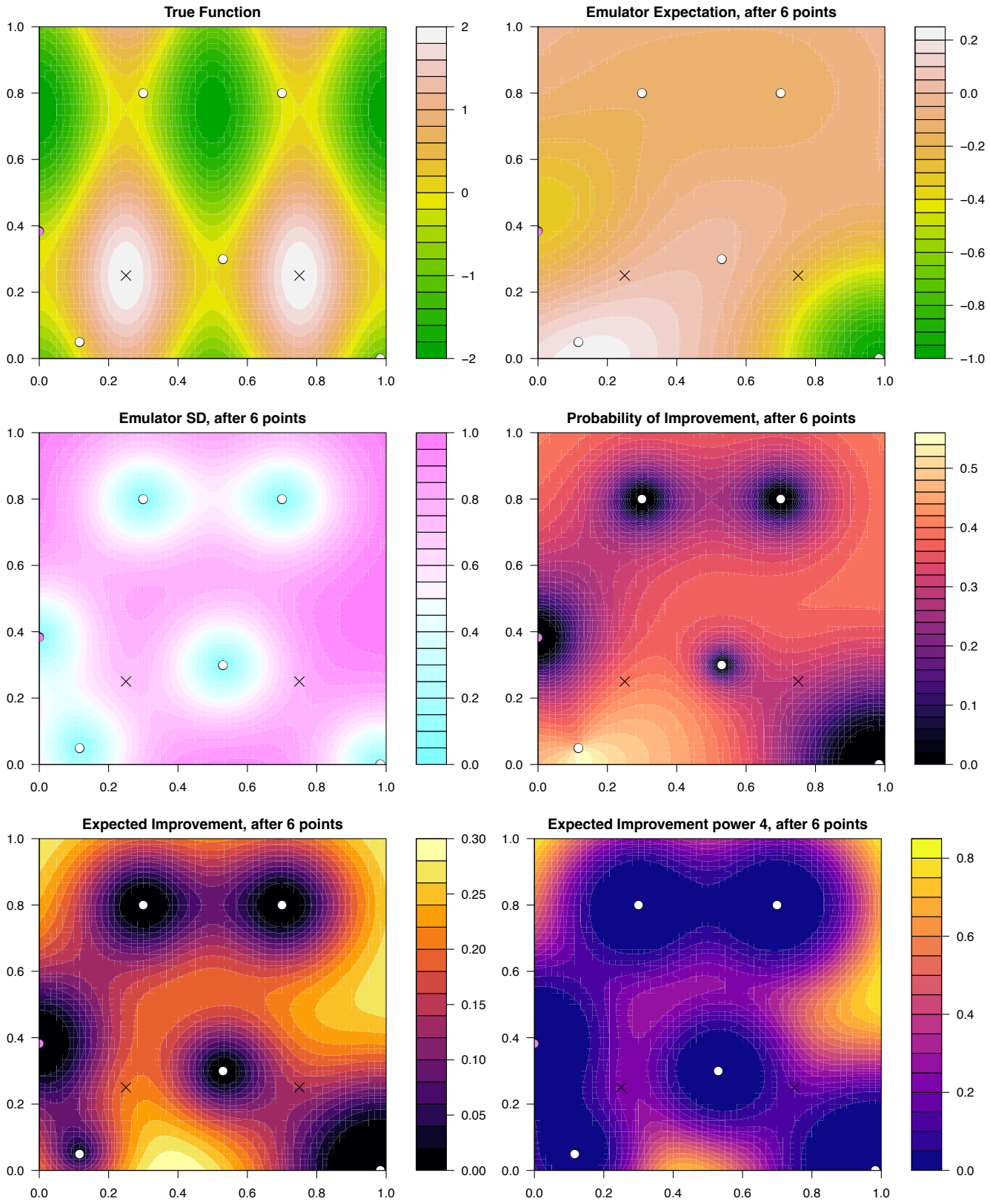


Figure 65: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

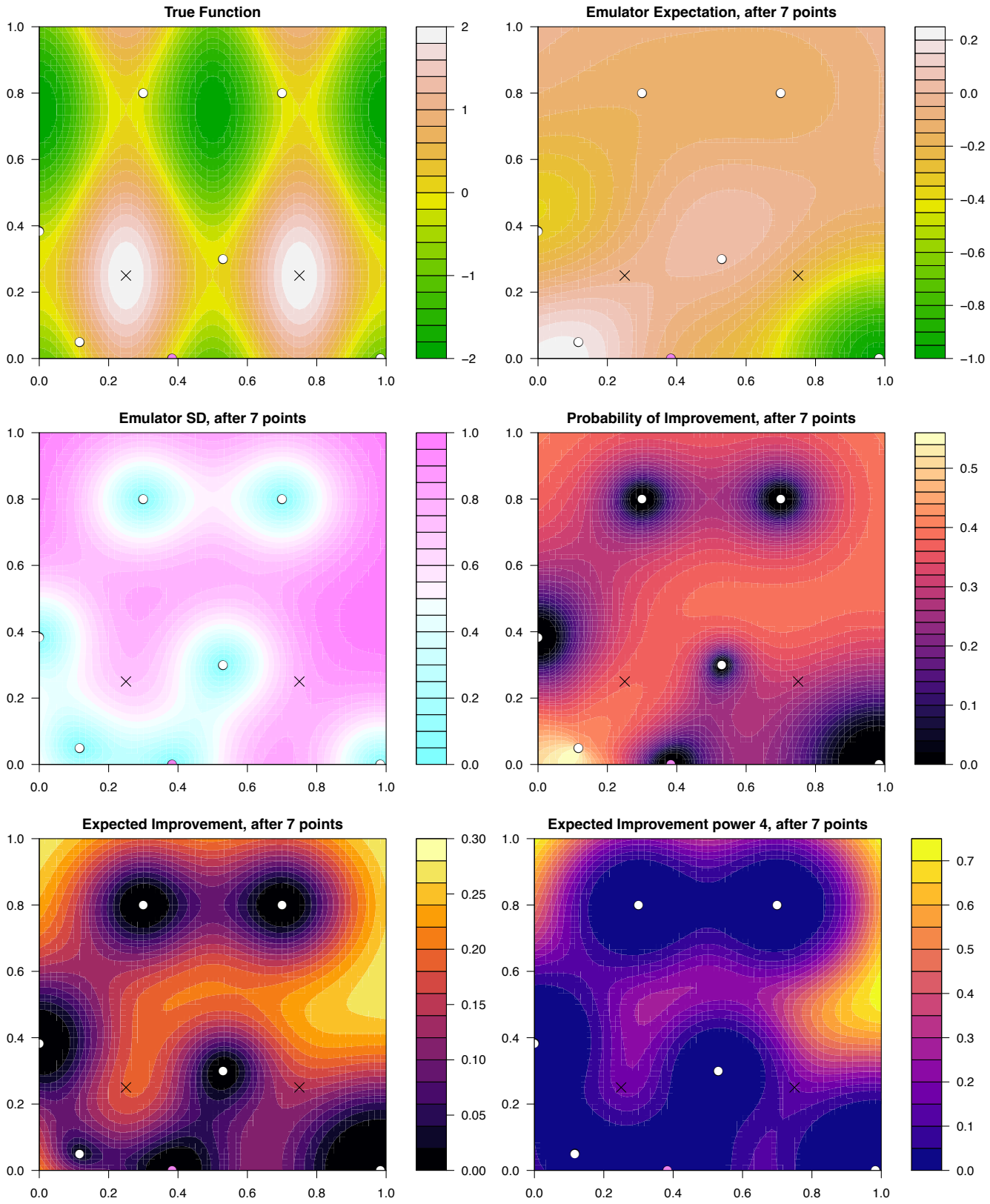


Figure 66: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

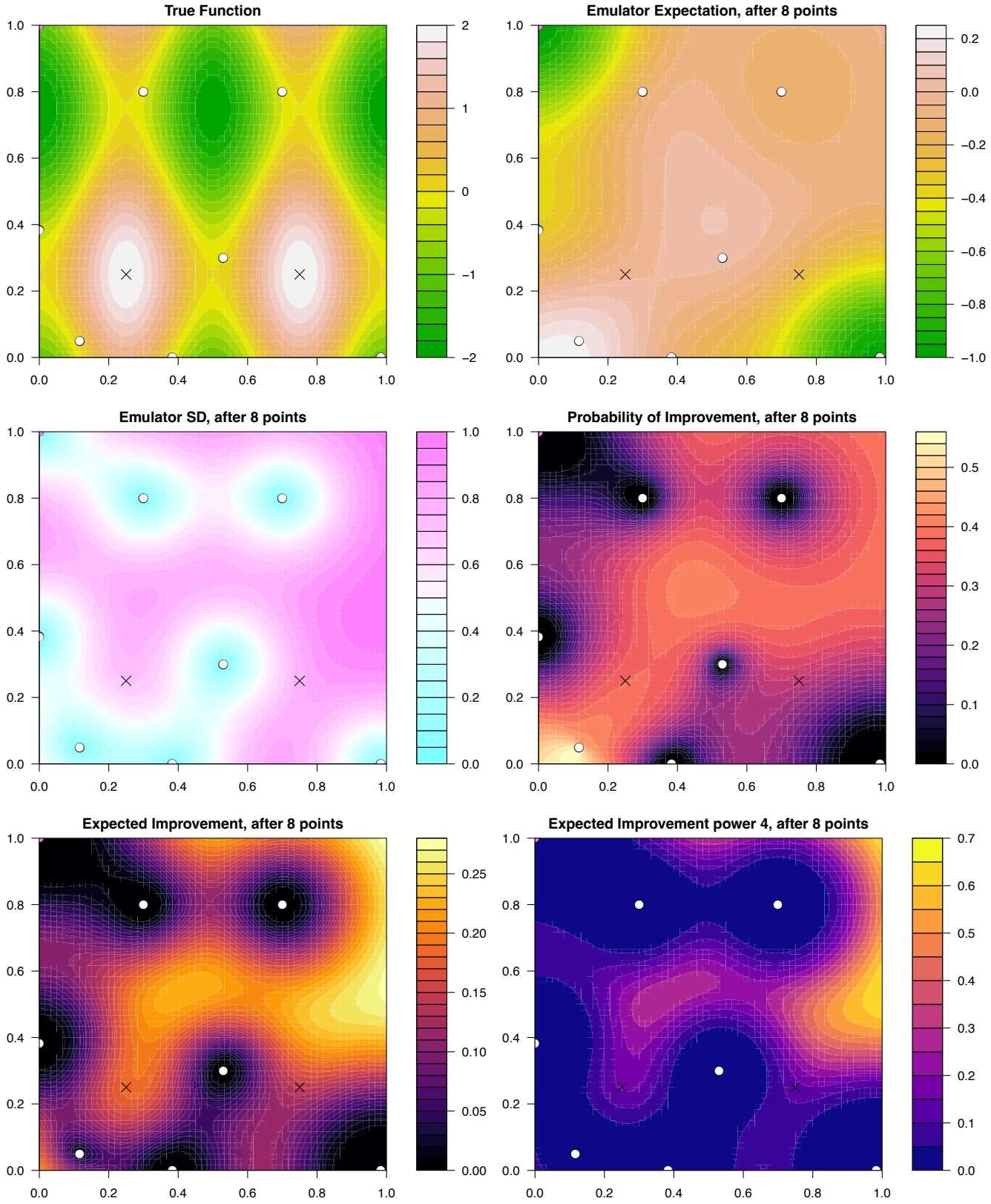


Figure 67: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

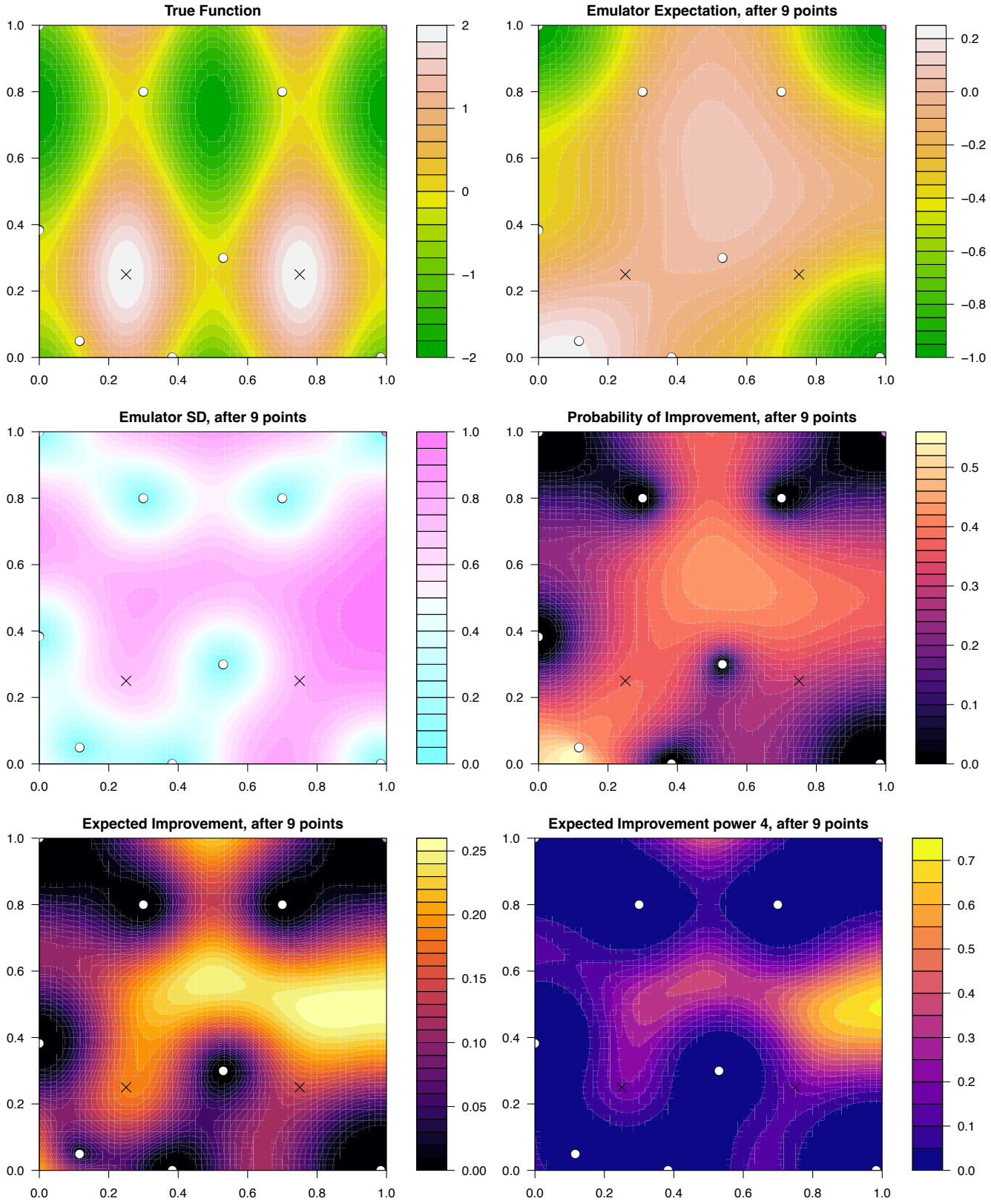


Figure 68: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

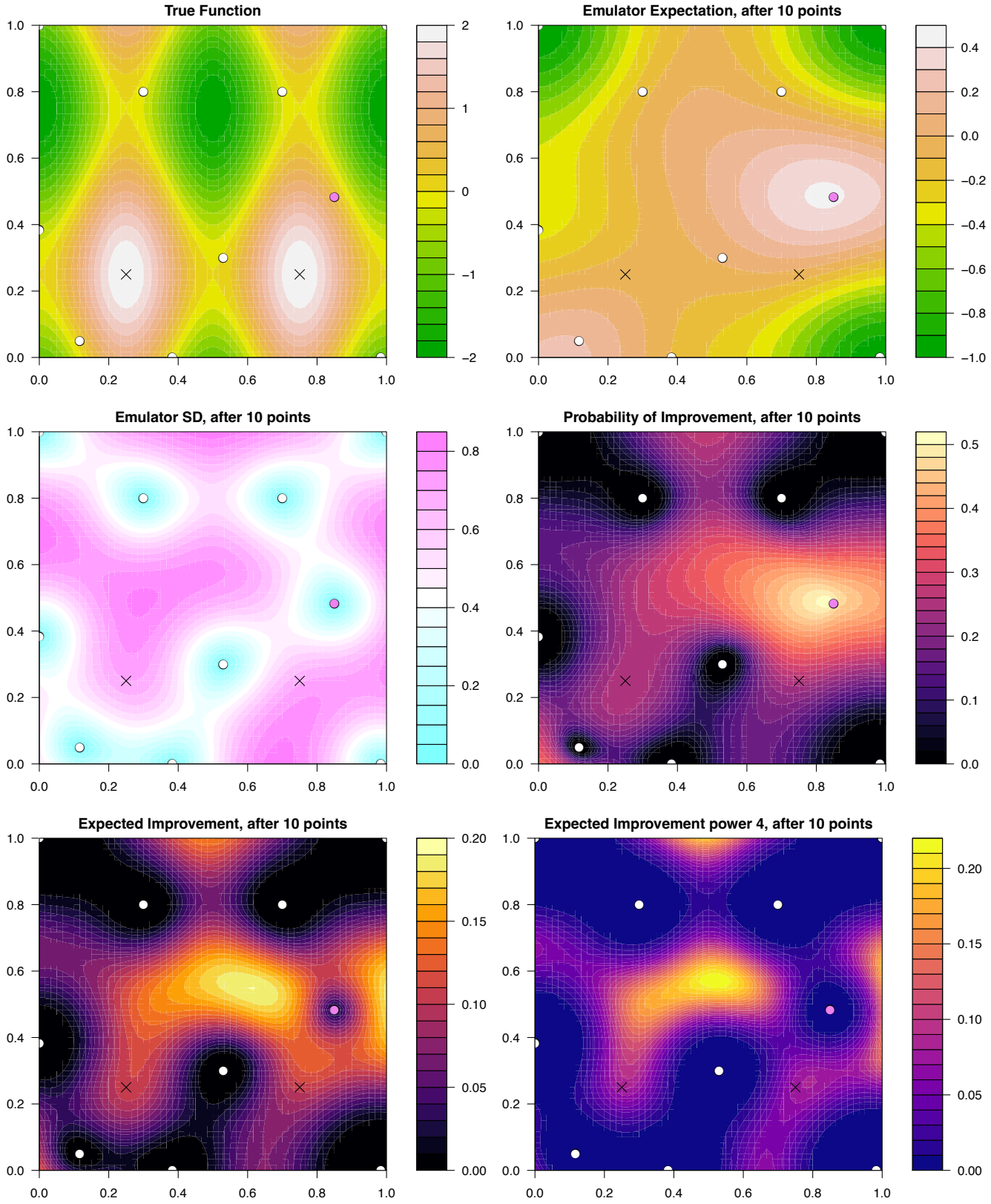


Figure 69: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

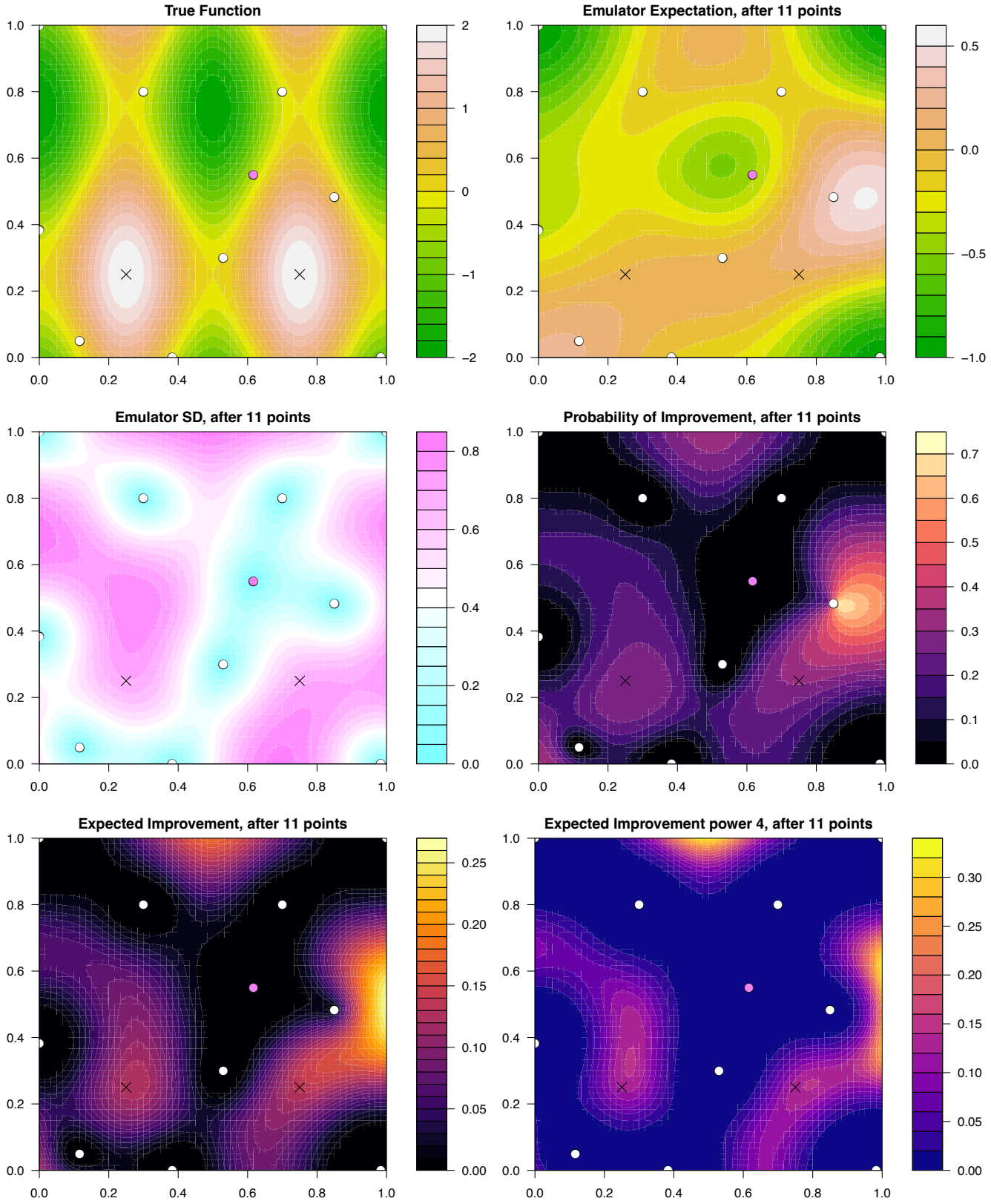


Figure 70: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

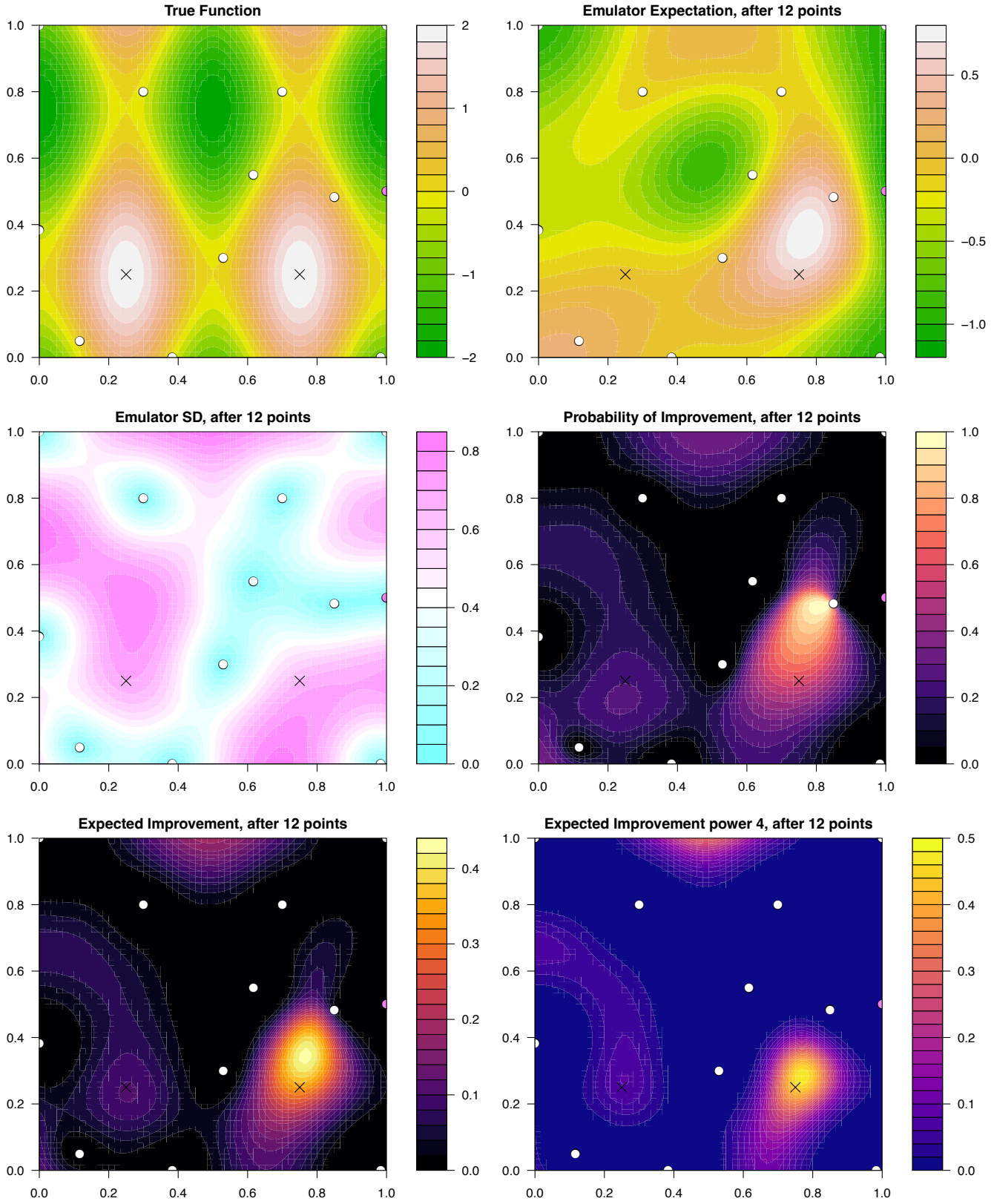


Figure 71: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

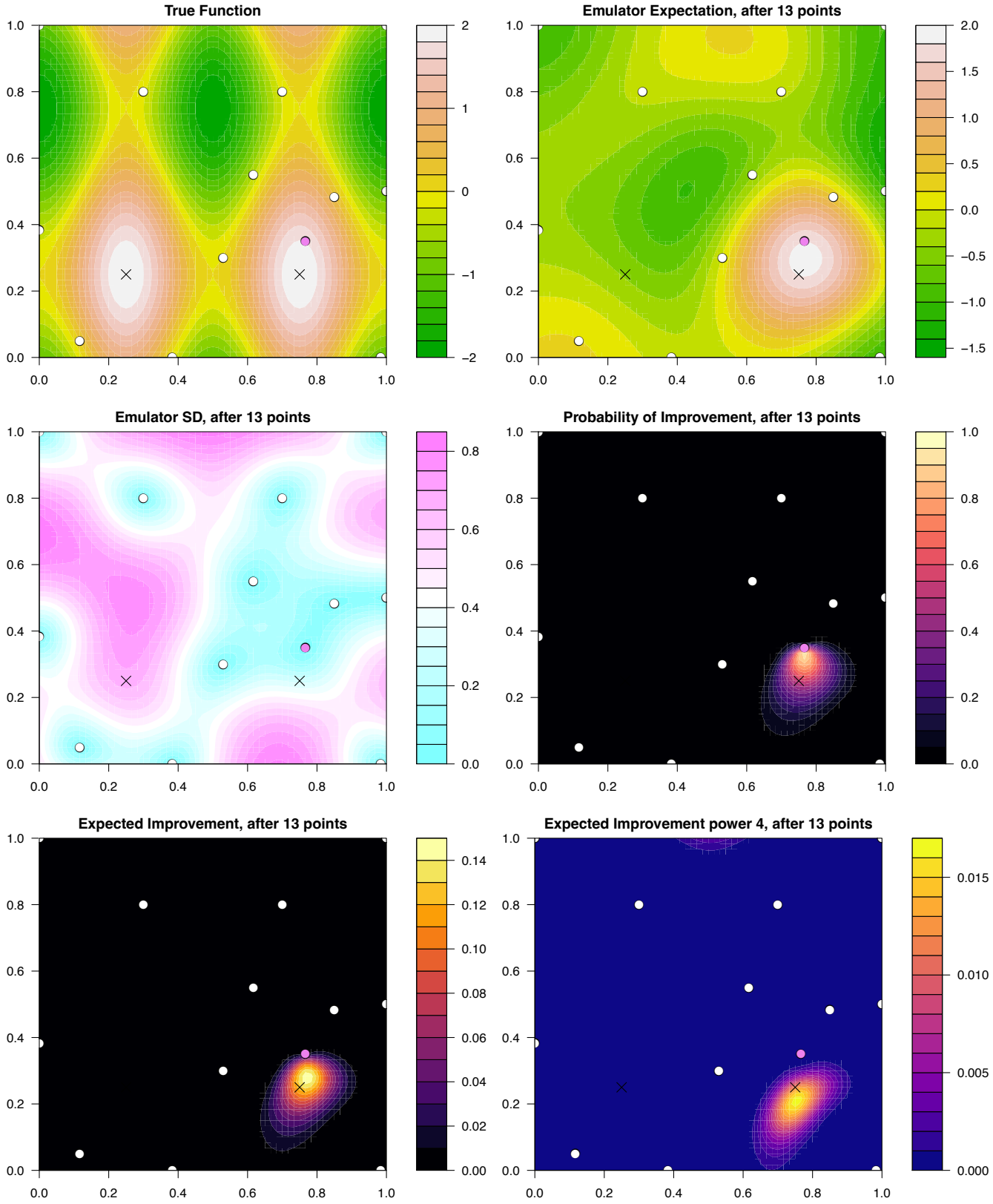


Figure 72: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

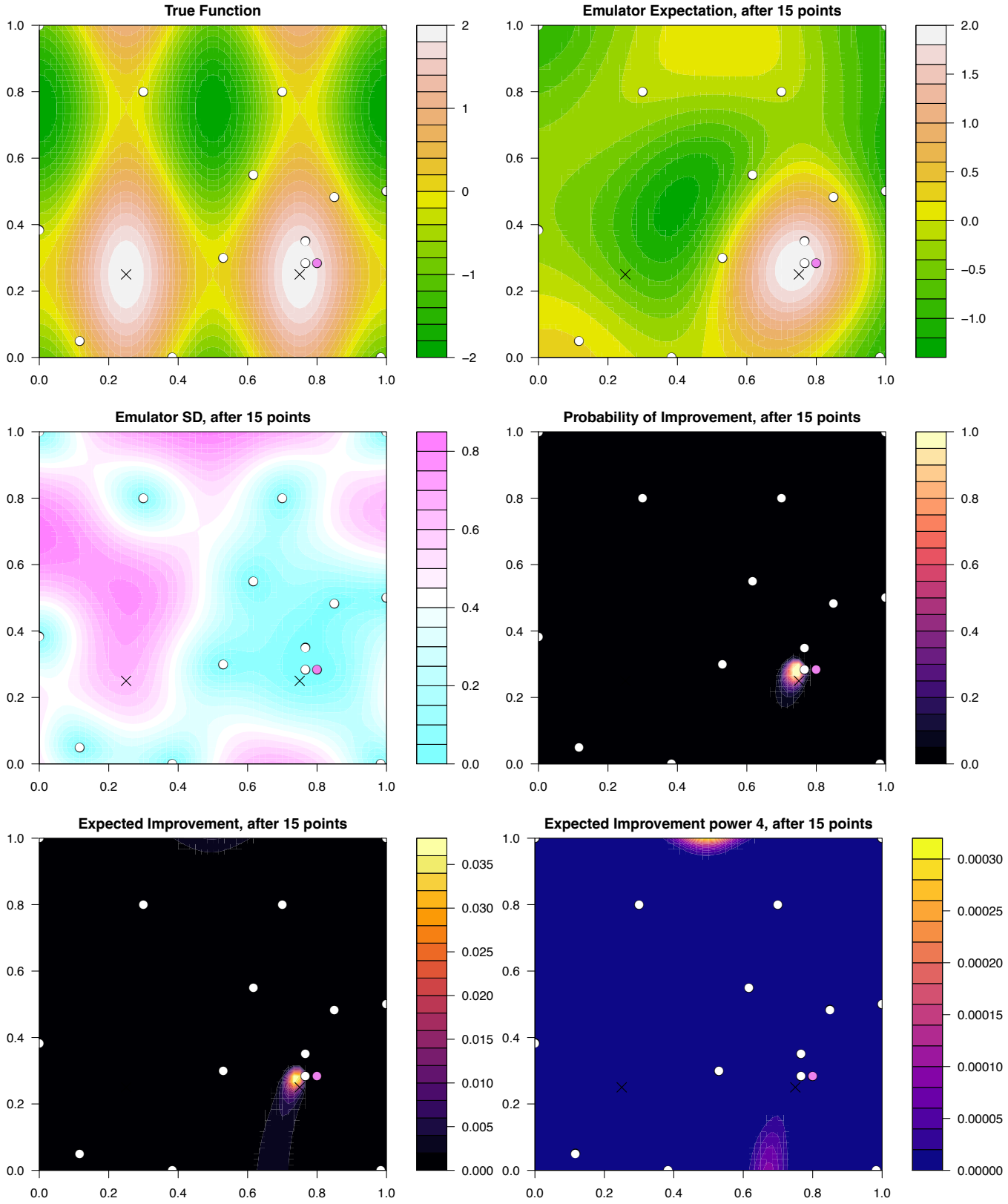


Figure 73: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

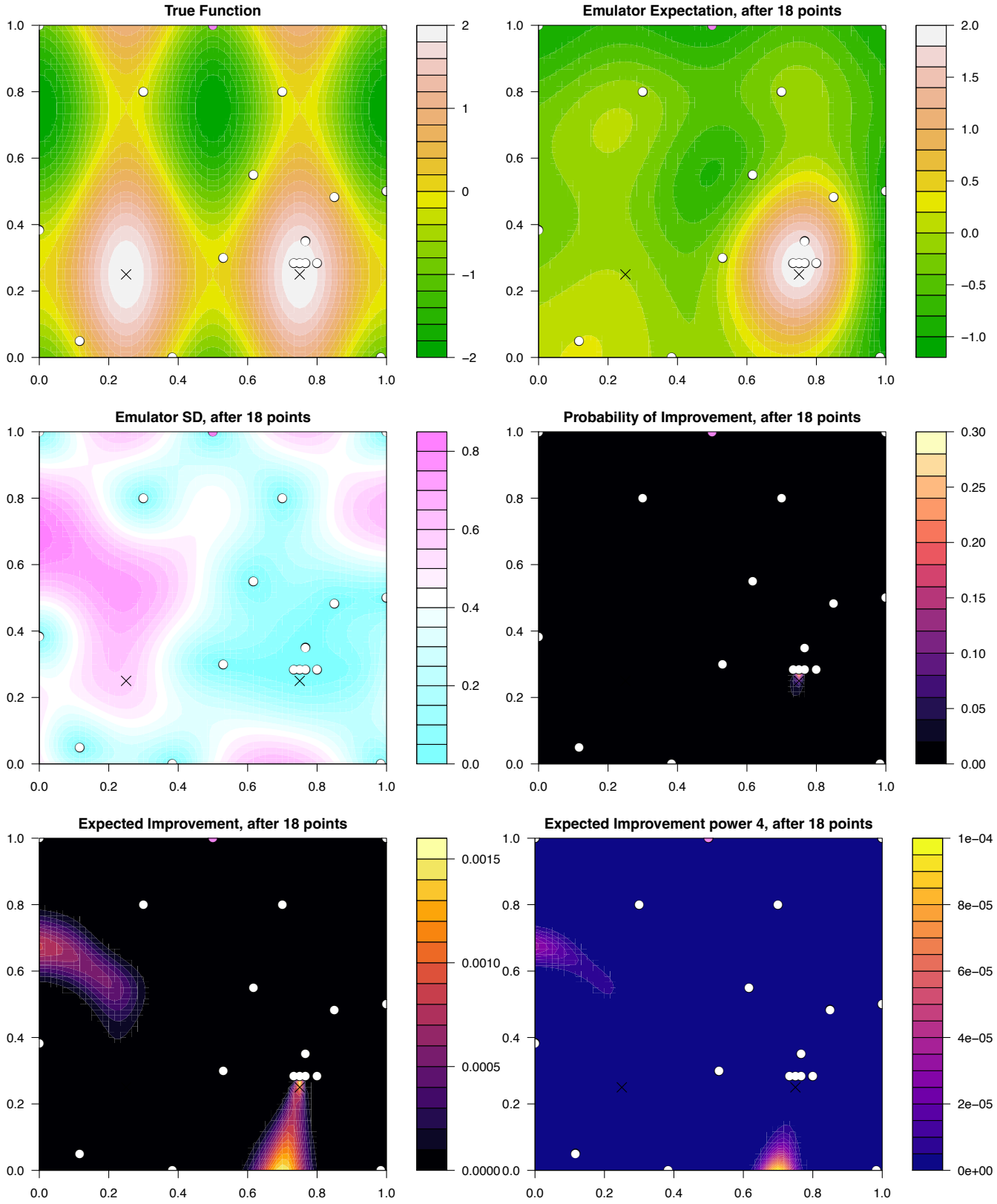


Figure 74: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

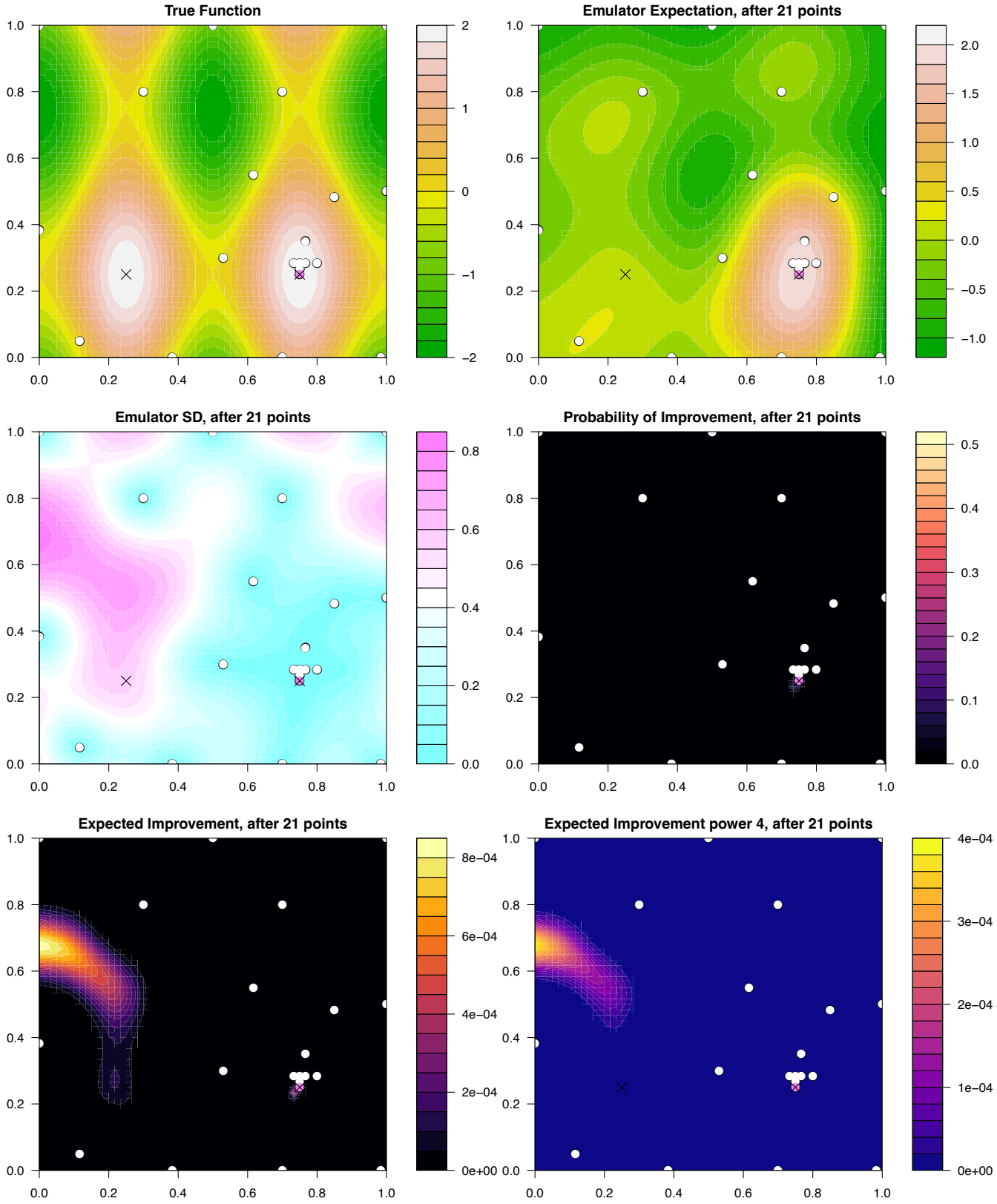


Figure 75: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

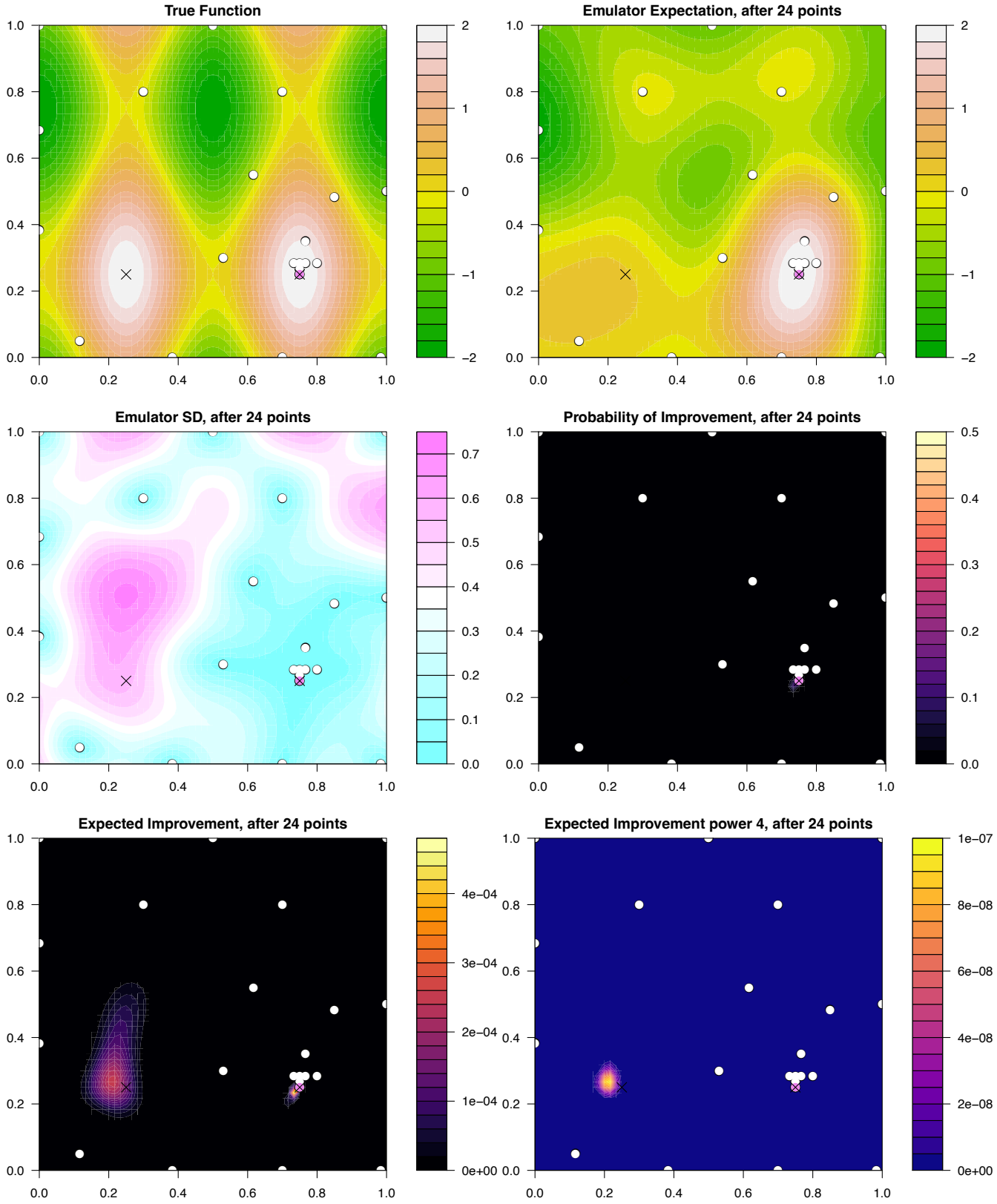


Figure 76: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

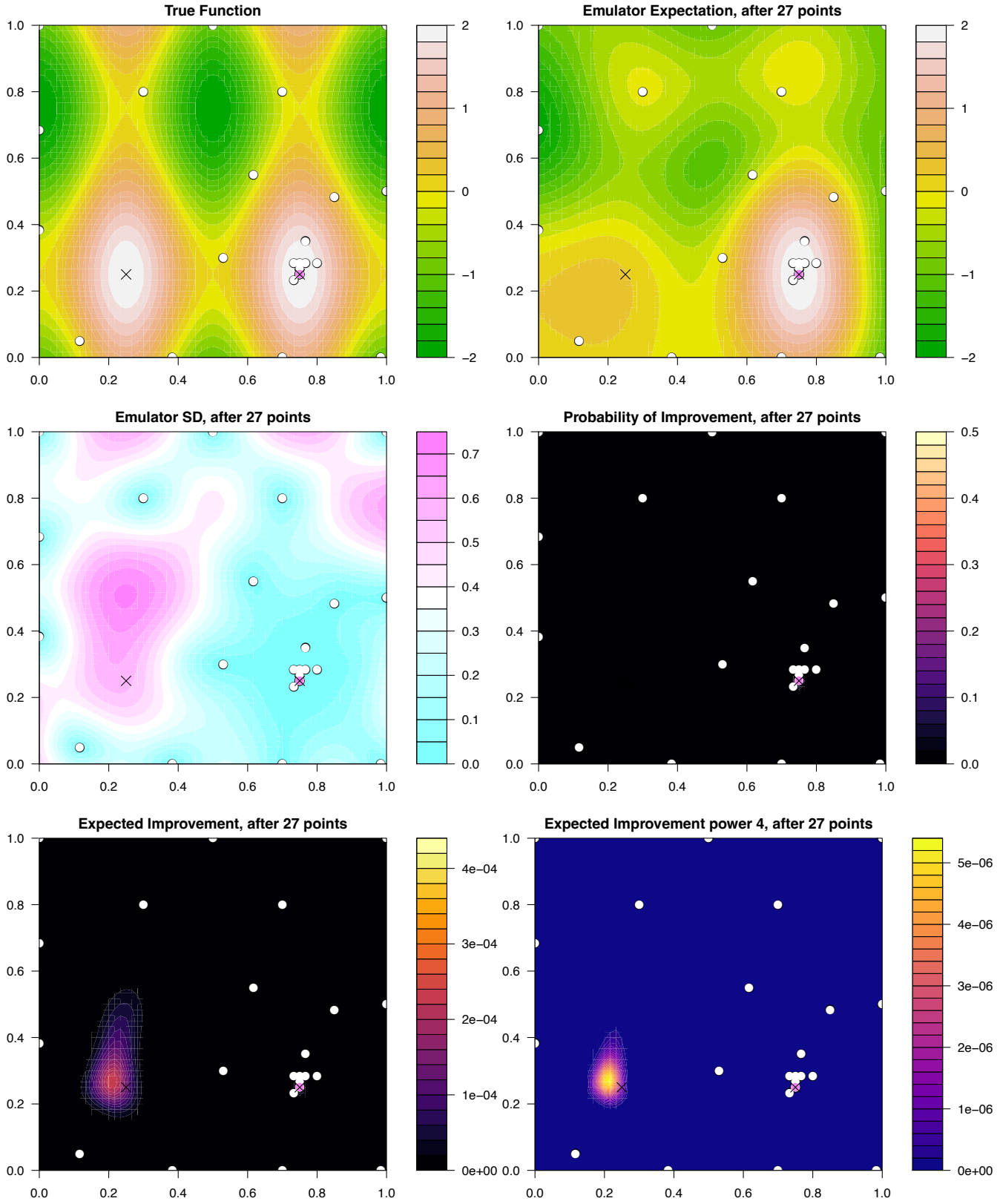


Figure 77: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

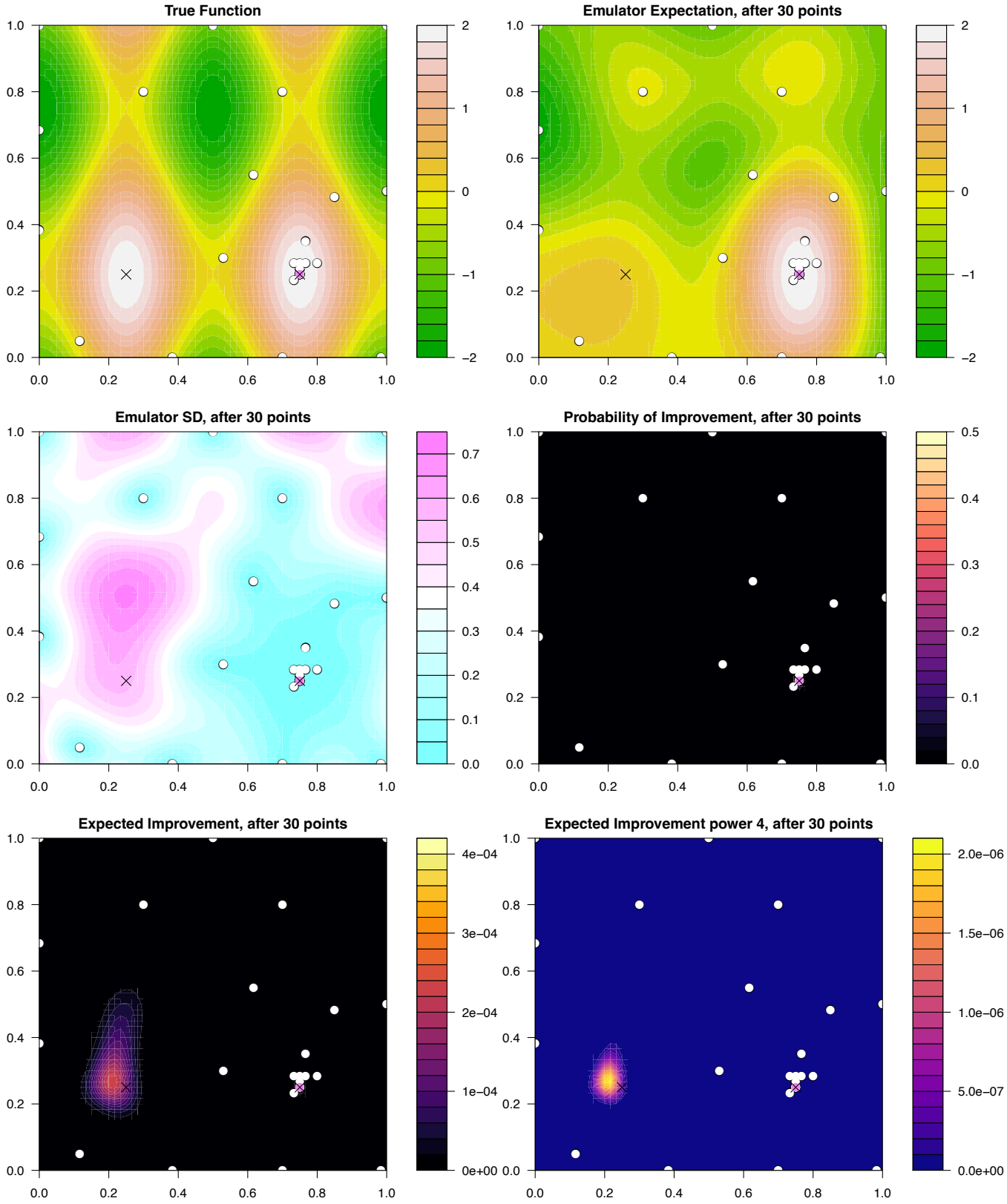


Figure 78: **Optimising using the $EI(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

7.3 Decision Theory Considerations

- We now map the two previously discussed criteria into a decision theory framework, in order to understand their structure, and to help suggest improvements/alterations.
- Decision Theory dictates that for any decision we simply seek to maximise our expected utility, either for a single decision, or for a sequential series of decisions. Our utility $U(x)$, combined with probabilities representing our beliefs about all future events is all that is required.
- For a single decision, where we will perform only one more run then stop, decision theory tells us to choose the single x value such that

$$x = \underset{x}{\operatorname{argmax}} \, \mathrm{E}[U(x)] \tag{153}$$

- This form is similar to that of the $PI(x)$ and $EI(x)$ approaches, which are both one step ahead/single decision/*myopic* approaches. We can hence use this to re-express the $PI(x)$ and $EI(x)$ approaches in terms of their underlying Utilities.

$PI(x)$ as a Utility

- For the $PI(x)$ criteria, we note that since a probability is equivalent to an expectation over an indicator function, we can rewrite $PI(x)$ in terms of a decision theory utility as follows:

$$\begin{aligned} PI(x) &= P(f(x) \geq f(x^+)) && (154) \\ &= \mathrm{E}[\mathbb{1}_{f(x)\geq f(x^+)}] && (155) \end{aligned}$$

  where we have used the indicator function $\mathbb{1}_A$ that takes value 1 when statement A is true and 0 when A is false:

$$\mathbb{1}_A = \begin{cases} 1\,, & A \text{ true} \\ 0\,, & A \text{ false} \end{cases} \tag{156}$$

- We see therefore that this $PI(x)$ criteria implicitly insists that our utility is just the indicator function of the improvement:

$$U_{PI}(x) = \mathbb{1}_{f(x)\geq f(x^+)} \tag{157}$$

  Remember that the Utility is the purest expression of what we actually care about, i.e. how we prioritise all the various possible outcomes of evaluating $f(x)$.

- Expressing the $PI(x)$ criteria in this Utility based form gives us insight into its weaknesses: it demands that as our utility is just the indicator function given in equation (157), we view all possible improvements as equally important, regardless of the magnitude of their improvements (and incidentally, all non-improvements as equally bad). Hence, this criteria prefers safe bets of say guaranteed incremental improvement, over modestly riskier bets that give a high probability of yielding dramatic improvements but with a small probability of no improvement.

- Hence $PI(x)$ often displays exploitation and not enough exploration, and is rightly now rarely used. It is only the correct criteria if a) we had only one more run to perform and b) it was vital that we simply beat the previous best run, no matter if it is only by a tiny amount.

- The adapted version $PI_\xi(x)$ has similar utility function:

$$U_{PI_\xi}(x) \quad = \quad \mathbb{1}_{f(x)\geq f(x^+)+\xi} \tag{158}$$

  and hence similar problems. While it now only favours improvements greater than ξ it still views these as all of equal worth. This procedure is naturally very sensitive to the choice of ξ and is not recommended. We need a far more refined statement of our desired outcomes.

$\underline{EI(x)}$ **as a Utility**

- We can also express the Expected Improvement criteria/acquisition function in terms of Utility, and similarly examine its implicit assumptions.

- As here we are told to find the x that maximises $\mathrm{E}I(x)$, the expectation of the improvement, matching this with equation (153) we trivially see that

$$U_{EI}(x) \quad = \quad \mathcal{I}(x) \quad = \quad \max\{0, f(x) - f(x^+)\} \tag{159}$$

  hence our utility is the linear improvement itself.

- We see that this utility is clearly more palatable than that of $PI(x)$, as it explicitly says that we prefer larger improvements, and depending on the various probabilities, we will use this attribute to perform some exploratory runs instead of the usual exploitation that dominates $PI(x)$.

- However we see that our utility here is linear in the improvement. It is not clear that this is the best choice. If we are interested in prioritising larger improvements (of lower probability), we may prefer a more risk-seeking utility function. The linear form was historically chosen mainly for mathematical convenience, although it does have good properties in certain simple settings.

- The linear form can of course be justified if we are performing one more run and we are indifferent between a) a guaranteed improvement of c, and b) a gamble that returns $2c$ or 0 with probability 0.5 for each, and all such similar pairs of gambles[1].
- If not, this raises the question, can we choose a more realistic utility function that reflects our preferences more accurately, and hence has in some sense superior performance to $EI(x)$?
- The additional (and critically important) question of sequential decision making, where we know we will choose several runs in a row, we discuss later.

7.4 Controlling the Exploration/Exploitation of Expected Improvement

- Treating the optimisation problem as a decision problem, as we have done above, we see that we are free to maximises the expectation of any function of the improvement $\mathcal{I}(x)$ (or even any function of $f(x)$), as long as we are prepared to identify this with our actual utility.
- If we prioritise larger (but less likely) improvements, we may construct more risk-seeking utility functions, that will exhibit more exploratory behaviour. A simple approach to achieve this is to raise the improvement to a power $q > 1$ (a choice of $q < 1$ would lead to more exploitation). Thus we define the $EI^q(x)$ criteria to have utility

$$U_{EI_q}(x) \;=\; \mathcal{I}(x)^q \;=\; \max\{0, (f(x) - f(x^+))^q\} \tag{160}$$

- A large value of q will have the effect of making the algorithm explore more, favouring places that could give larger gains even for lower probability, and getting stuck less often next to good runs.
- This approach may lead to an emulator that is more accurate over all the parts of input space of interest (i.e. all the parts that have high values). This may give us more confidence that we have identified all possible modes, and hence provide a more robust approach.
- However, this will come at a cost: such an algorithm will be less inclined to aggressively optimise to find the precise peak of a particular mode, and may spend too much time exploring boundaries/corners.
- A natural choice would be $q = 2$, giving

$$U_{EI_2}(x) \;=\; \mathcal{I}(x)^2 \tag{161}$$

[1] Technically, we need to be indifferent between a) a guaranteed improvement of c, and b) a gamble that returns c/p or 0 with probabilities p and $1 - p$ respectively, where $0 \leq p \leq 1$.

Again we can derive the corresponding analytic expression for the expectation of $U_{EI_2}(x)$ (see problem sheet):

$$\mathrm{E}[U_{EI_2}(x)] \;=\; \sigma_D(x)c(x)\phi(z^*) + (\sigma_D^2(x) + c^2(x))\Phi(z^*) \tag{162}$$

where again $z^* = (\mu_D(x) - f^+)/\sigma_D(x)$ and here we define the deviation $c(x) = \mu_D(x) - f^+$. The second term shows, for example, that for large positive values of z^*, for which $\phi(z^*) \simeq 0$ and $\Phi(z^*) \simeq 1$, we have that

$$z^* \gg 1 \quad \Rightarrow \quad \mathrm{E}[U_{EI_2}(x)] \;\simeq\; \sigma_D^2(x) + c^2(x) \tag{163}$$

and hence the utility will be dominated equally by large variances (i.e. emulator uncertainties) and by large predicted deviances squared, which may be a desirable balance. For values of $c(x)$ (and hence z^*) close to zero, which occurs when the emulator mean $\mu_D(x)$ is close to the best run output f^+, we see the behaviour is now

$$z^* \simeq 0 \quad \Rightarrow \quad \mathrm{E}[U_{EI_2}(x)] \;\simeq\; \frac{1}{2}\sigma_D^2(x) \tag{164}$$

with a quadratic dependence on $\sigma_D(x)$ that will disfavour runs with low $\sigma_D(x)$ such as the incremental improvements favoured by $PI(x)$, in favour of more uncertain locations.

- The expression for $\mathrm{E}[U_{EI_2}(x)]$ given by equation (162) should be compared to the standard $EI(x)$ formula given by equation (150) which we rewrite here using $c(x)$:

$$EI(x) \;=\; \mathrm{E}[U_{EI}(x)] \;=\; \mathrm{E}[\mathcal{I}(x)] \;=\; \sigma_D(x)\phi(z^*) + c(x)\Phi(z^*), \tag{165}$$

where, for example the large and positive z^* behaviour only depends on $c(x)$ and not on the emulator's uncertainty $\sigma_D(x)$, and the z^* close to zero behaviour depends linearly on $\sigma_D(x)$:

$$z^* \gg 1 \quad \Rightarrow \quad \mathrm{E}[U_{EI}(x)] \;\simeq\; c(x) \tag{166}$$

$$z^* \simeq 0 \quad \Rightarrow \quad \mathrm{E}[U_{EI}(x)] \;\simeq\; \frac{\sigma_D(x)}{\sqrt{2\pi}} \tag{167}$$

where the $\sqrt{2\pi}$ comes from the $\phi(0)$ term.

- Setting $q = 2$ is of course just one possible choice. We can in principle analytically calculate $\mathrm{E}[U_{EI_q}(x)]$ for any positive integer value of q, however for non-integer values (which do not in general have an analytic solution) or indeed if we wish to use more complex functions of $f(x)$ instead of $\mathcal{I}(x)^q$, we can use Monte Carlo estimates to solve the 1D integral required to calculate the relevant expectation, as 1D integrals are relatively cheap to numerically evaluate.

- As an exemplar, and to provide insight into the behaviour of a more exploratory focussed utility function, we will focus on the $q = 4$ case

$$U_{EI_4}(x) \;=\; I(x)^4 \tag{168}$$

and compare its performance to that of $EI(x)$ and also $PI(x)$.

Bayesian Optimisation: 1D Example Revisited with $U_{EI_4}(x)$

- The three plots in the right column of Figures 30 to 44 show the optimisation of the 1d example introduced in section 7.2, but now we use the $EI^4(x) = \mathrm{E}[U_{EI_4}(x)]$ acquisition function to choose the next run at each iteration.

- The $EI^4(x)$ acquisition function is shown as the green line in the bottom right panel, and the new input point x is shown as the vertical blue line, with previous run locations shown as vertical grey lines. The pink line gives the standard $EI(x)$ acquisition function as before. The top right (emulator predictions) and middle right ($PI(x)$ criteria) are as described in section 7.2.

- As expected, we see that the $EI^4(x)$ criteria prefers points that have greater uncertainty over that of $EI(x)$. In figure 30 we choose a point (at approximately $x = 2.3$) which is further away from the known run at $x = 1$, but only modestly further away than the $EI(x)$ choice. A real difference is seen in figure 34, where the $EI(x)$ criteria is now chasing a possible maximum and would put a point at $x = 7.8$ (bottom right panel, maximum pink peak). The $EI^4(x)$ criteria on the other hand, thinks that it would be too early to chase this maximum, and prefers to explore a high uncertainty point at $x = 4.2$. Intuitively, the $EI^4(x)$ approach may be satisfactory: we may think it is wise to pin down the highly uncertain regions, before starting to chase the maxima.

- This intuition is of course entangled with the knowledge that we can perform more than one evaluation. If we can only perform a single run, then we may well wish to target one of the maxima. This issue will be discussion shortly.

- We see similar deviations between the $EI^4(x)$ and $EI(x)$ approaches in figures 38, 39 and 40. Both approaches then chase the true maximum from figure 42 onwards, however we see that the emulator variance over the three peaks in the $EI^4(x)$ case is noticeably smaller as the runs are more spread out around the peaks and more exploration has been done to rule out the first (lowest) peak. We may view this as a more satisfactory outcome, and have higher confidence that we have indeed found the true global maximum.

Bayesian Optimisation: 2D Example Revisited with $U_{EI_4}(x)$

- Figures 79 to 95 show the optimisation of the 2d example introduced in section 7.2 but we now use the $EI^4(x) = \mathrm{E}[U_{EI_4}(x)]$ acquisition function to choose the next run at each iteration.
- The six plots in each figure are defined as before, with the $EI^4(x)$ acquisition function shown in the bottom right panel and the standard $EI(x)$ acquisition function in the bottom left panel. We also have the true function $f(x)$ and emulator mean $\mu_D(x)$ (top row), and the emulator standard deviation $\sigma_D(x)$ and $PI(x)$ criteria (middle row).
- If we flick through figures 79 to 88 while observing the plot of $EI^4(x)$ in the bottom right panel, we see that this criteria chooses the first 9 points in a systematic symmetric space filling way, targeting points with high uncertainty mainly in the corners and on the boundaries of the search space.
- It then targets the right of centre peak, and uses several points first to explore and then to optimise this peak (figures 89 to 92).
- Then, the region of uncertainty around the left of centre peak is noticed (figure 92), shown as the pink region in the emulator SD plot, left middle panel. The criteria then lands points in this region, realises that it contains high values, and subsequently optimises this left of centre peak (figures 93 to 94).
- For the final couple of runs the $EI^4(x)$ criteria jumps back and forth between the two global maxima (figures 94 and 95).
- Comparing with the behaviour of the standard $EI(x)$ criteria as shown in figures 62 to 78, we see that for this example the $EI^4(x)$ provides an improved performance in that it finds and optimises the right peak using similar numbers of runs to the $EI(x)$ criteria, but then locates and optimises the second peak, while the $EI(x)$ criteria remains stuck on the right peak.
- We emphasise that this is just one example, which we are using to highlight the various features and weaknesses of these types of algorithms.
- A final point about the $EI^4(x)$ criteria: as we have seen, at early stages it puts points in places of high emulator uncertainty which are often on boundaries and at corners. This can be problematic in high dimensions where there are a lot of corners (in m dimensions there are 2^m corners for example). This will be somewhat mitigated by the form of the emulator: e.g. using global regression terms as described in section 5 controls the variance at the boundaries to a certain extent, and may limit this issue.

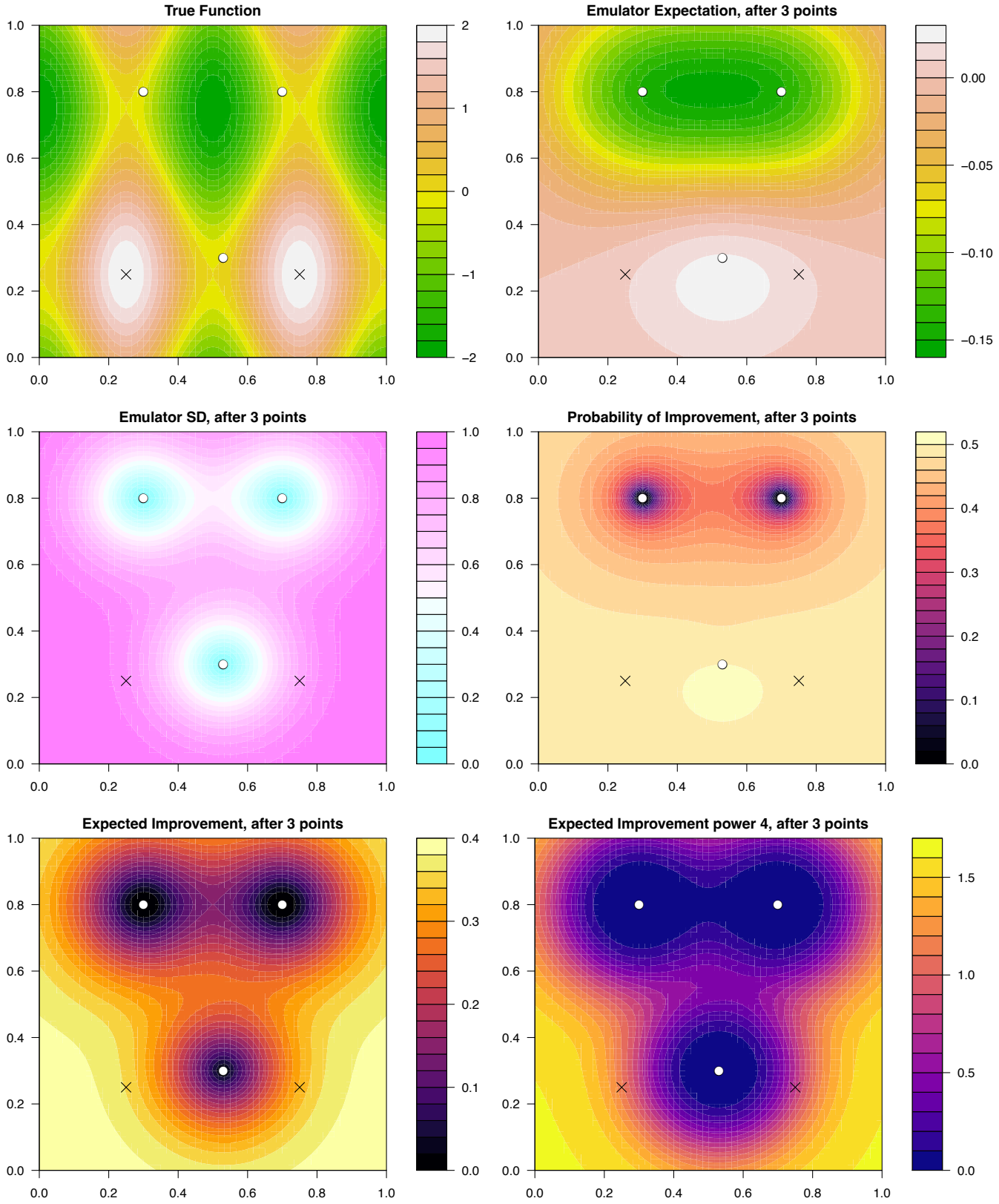


Figure 79: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

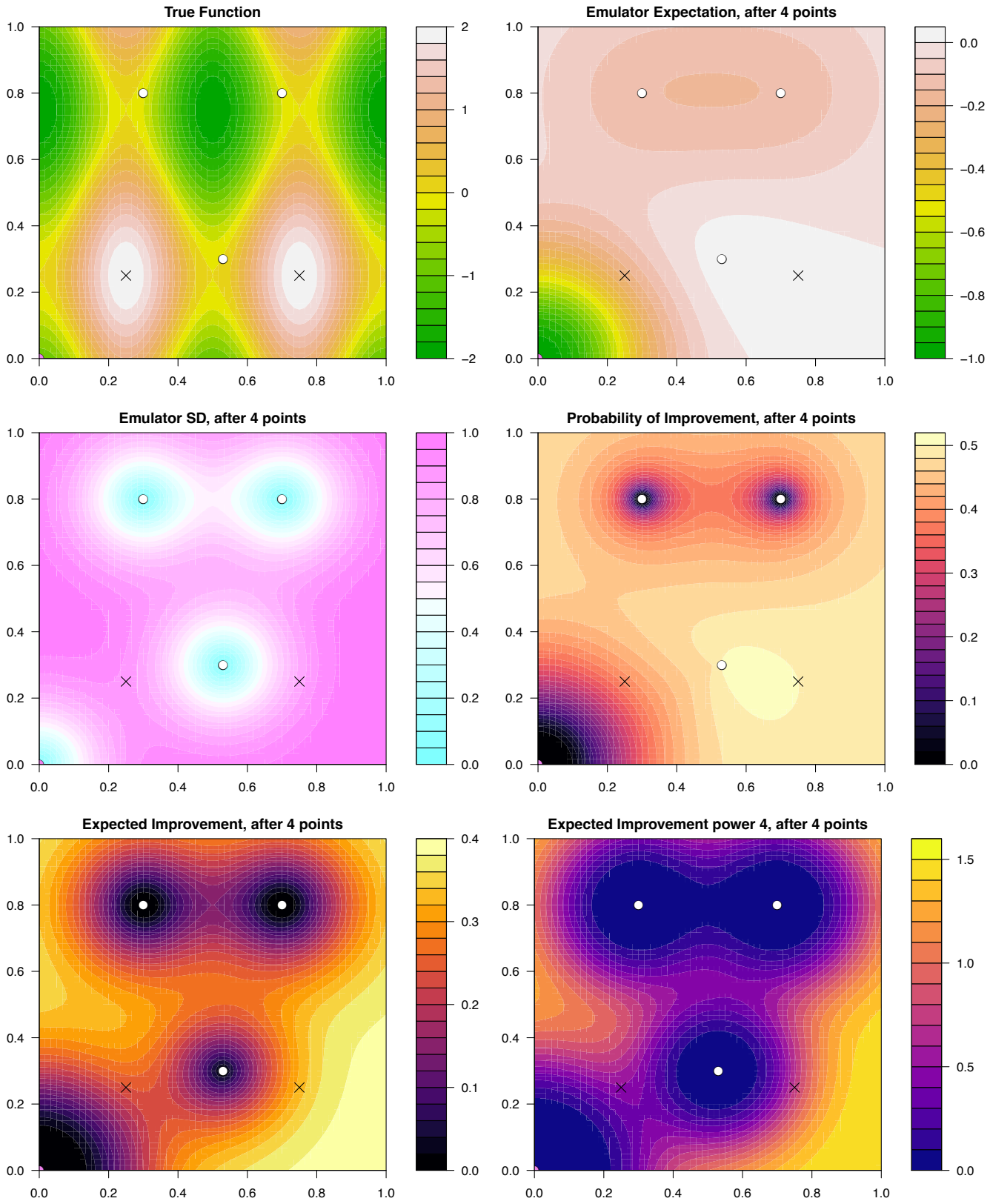


Figure 80: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

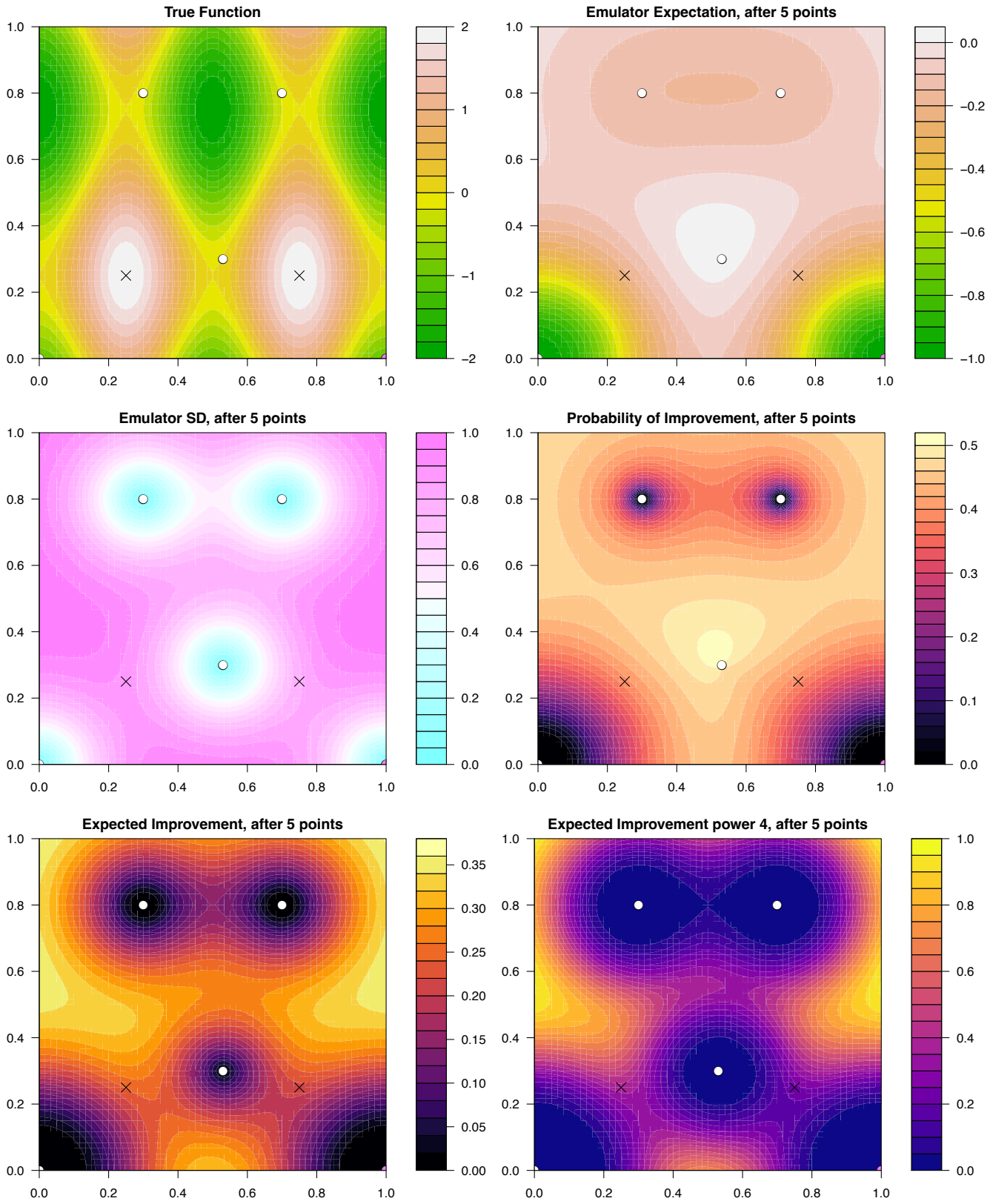


Figure 81: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

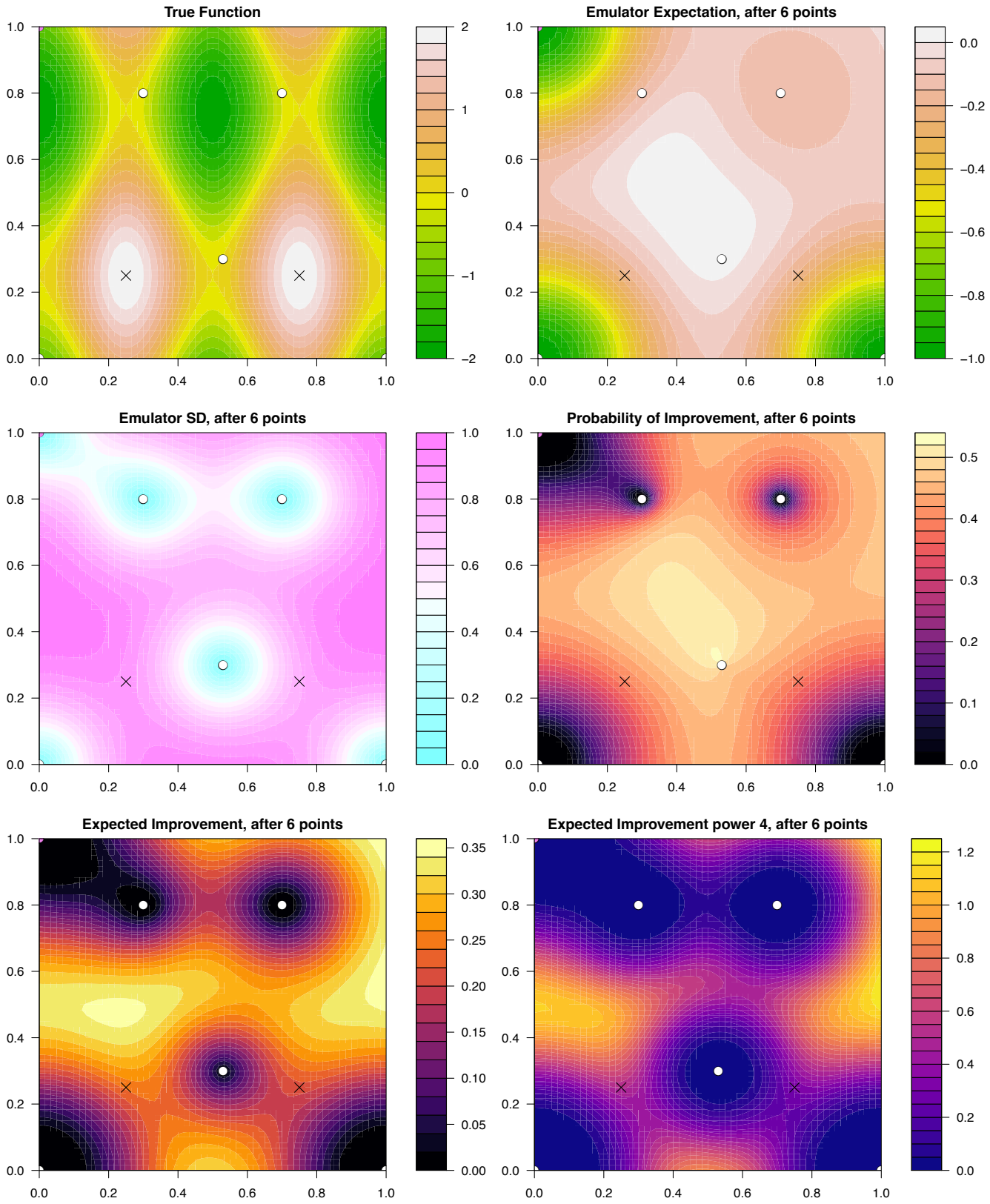


Figure 82: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

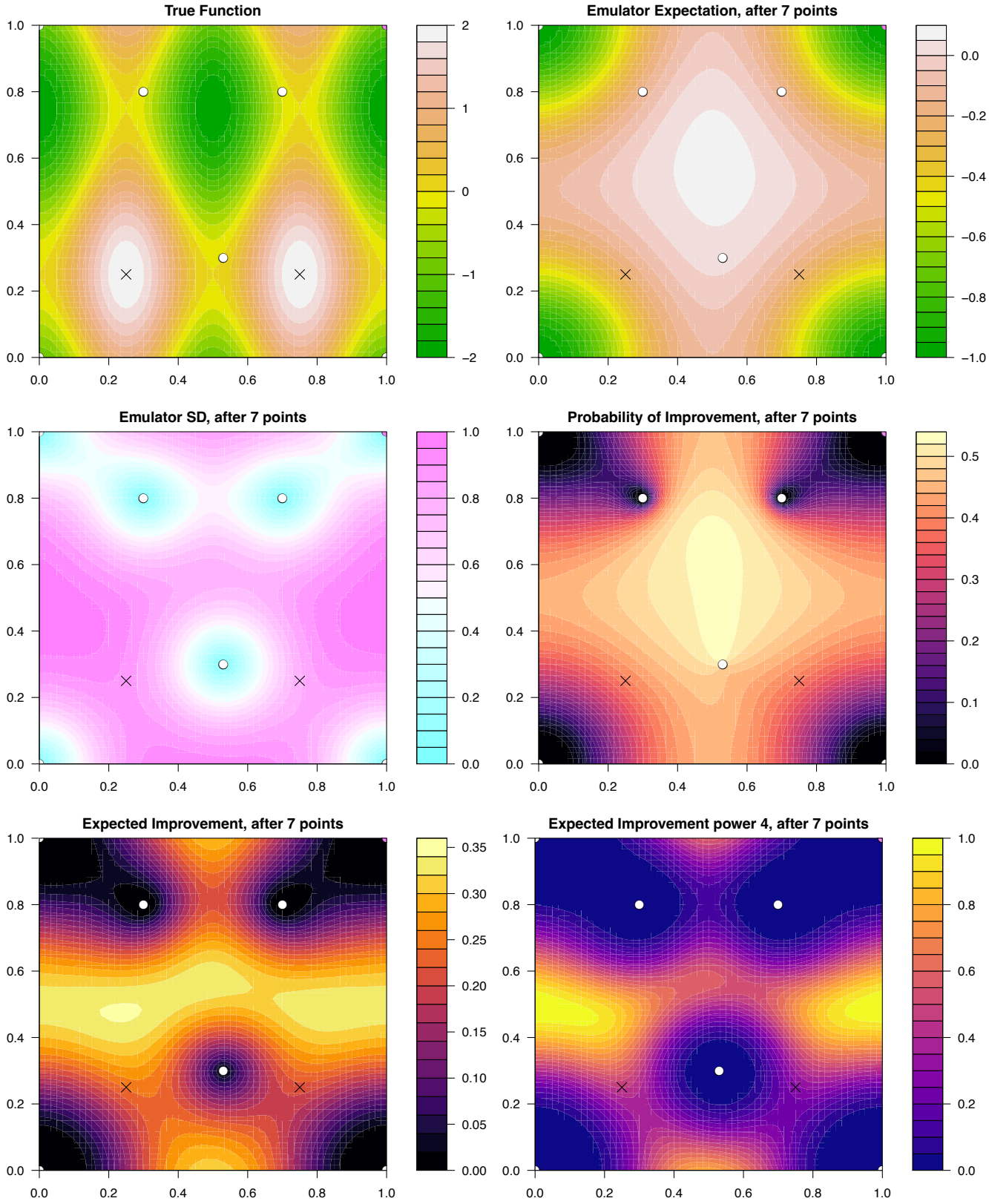


Figure 83: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

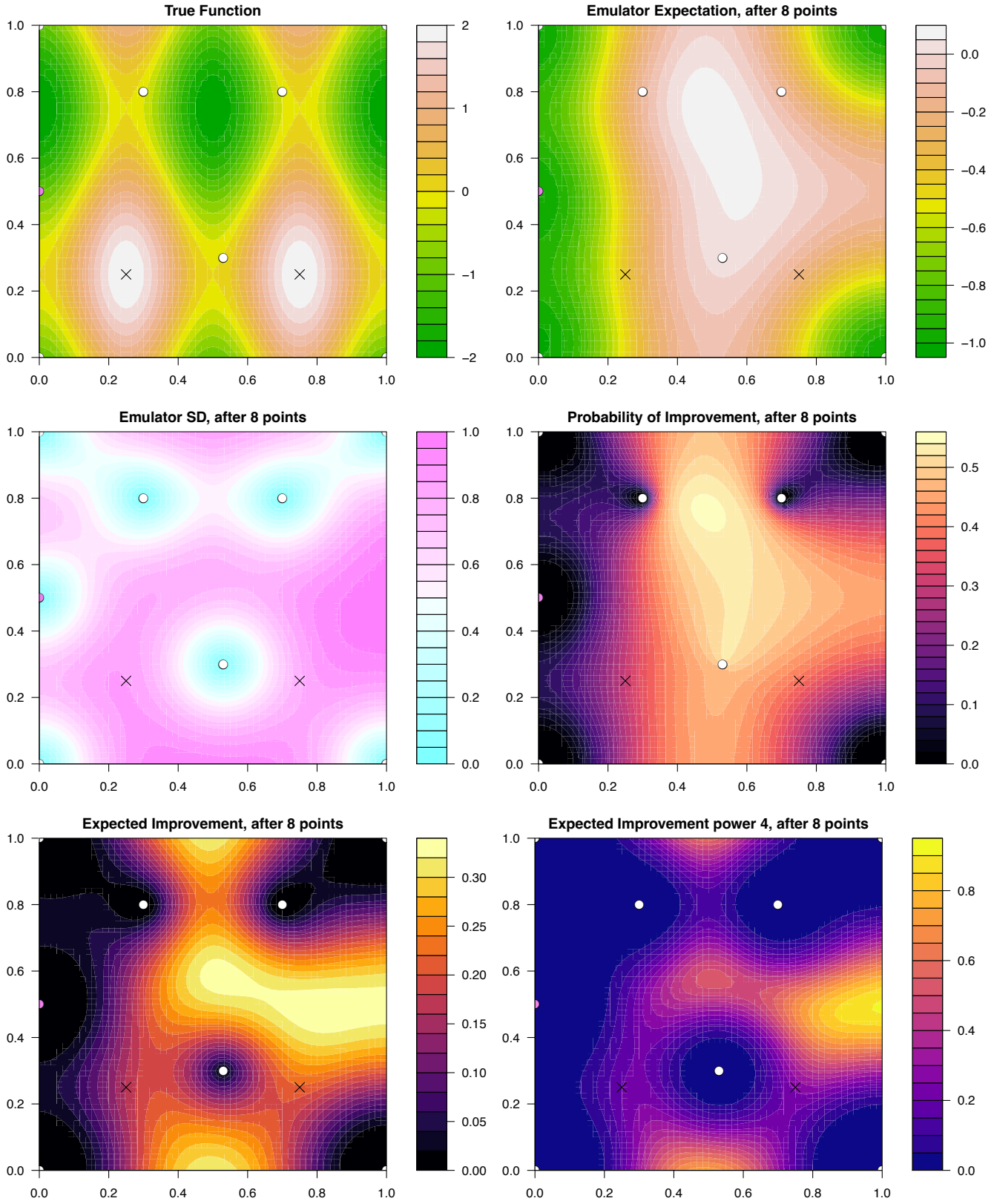


Figure 84: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

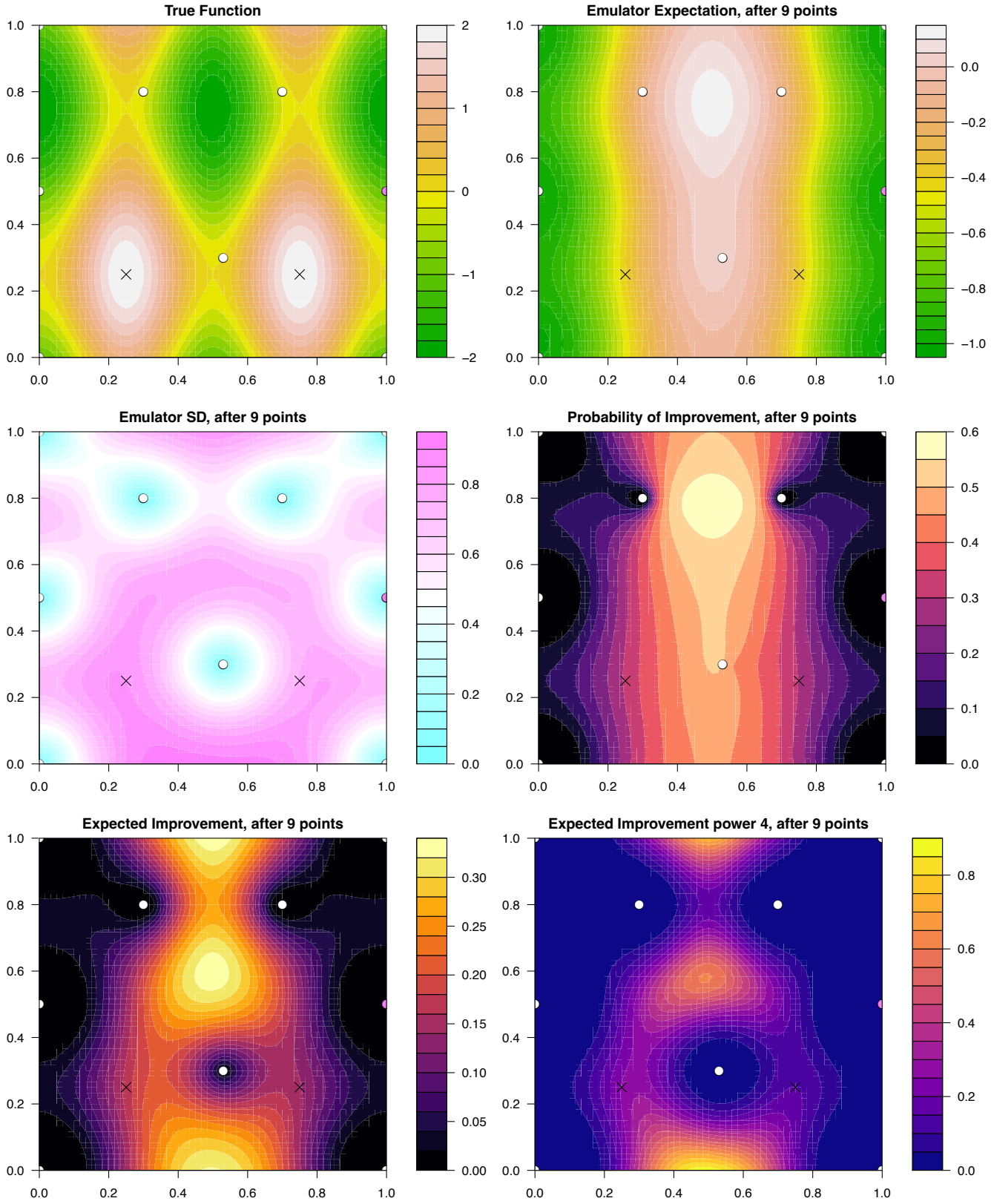


Figure 85: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

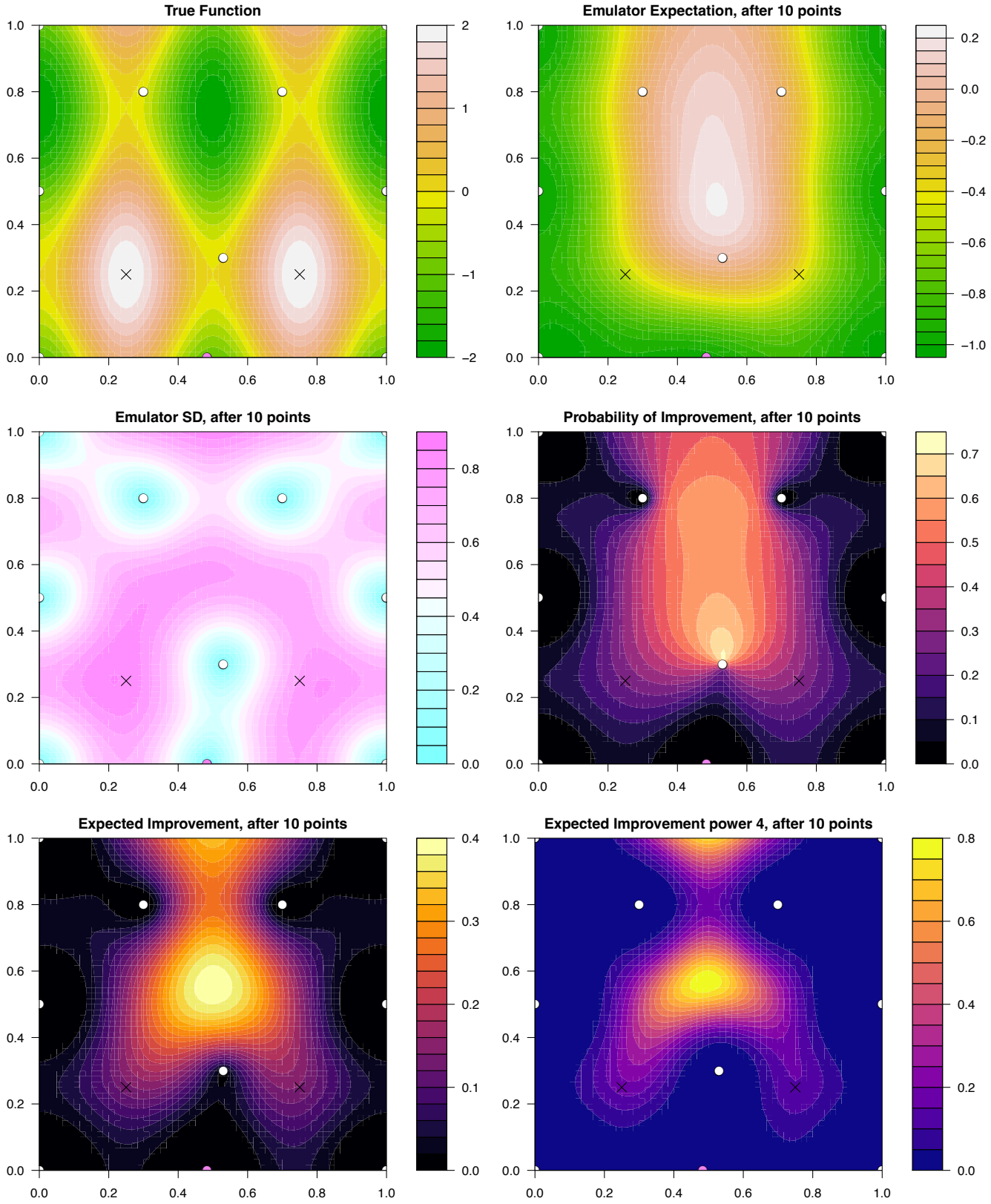


Figure 86: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

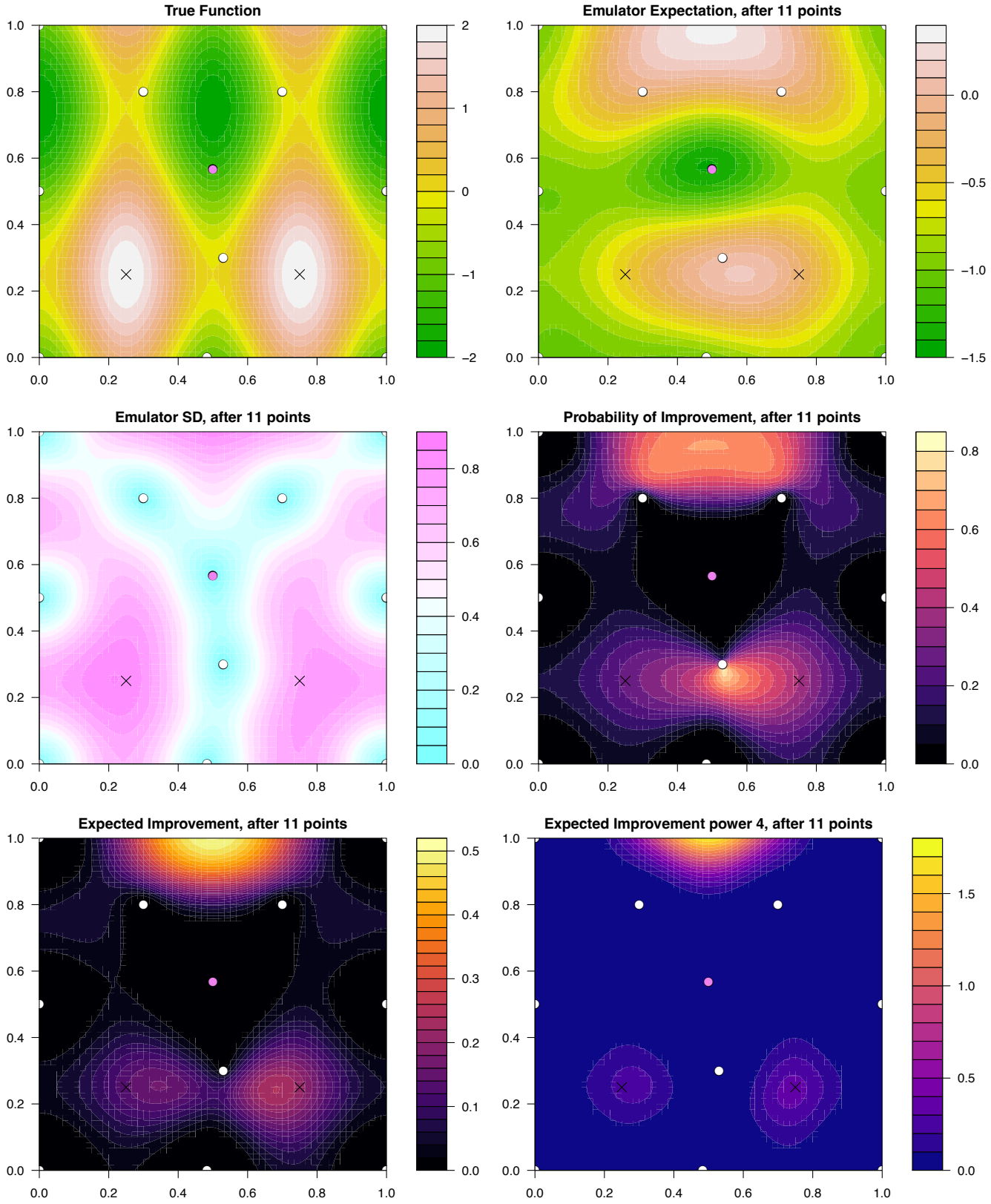


Figure 87: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

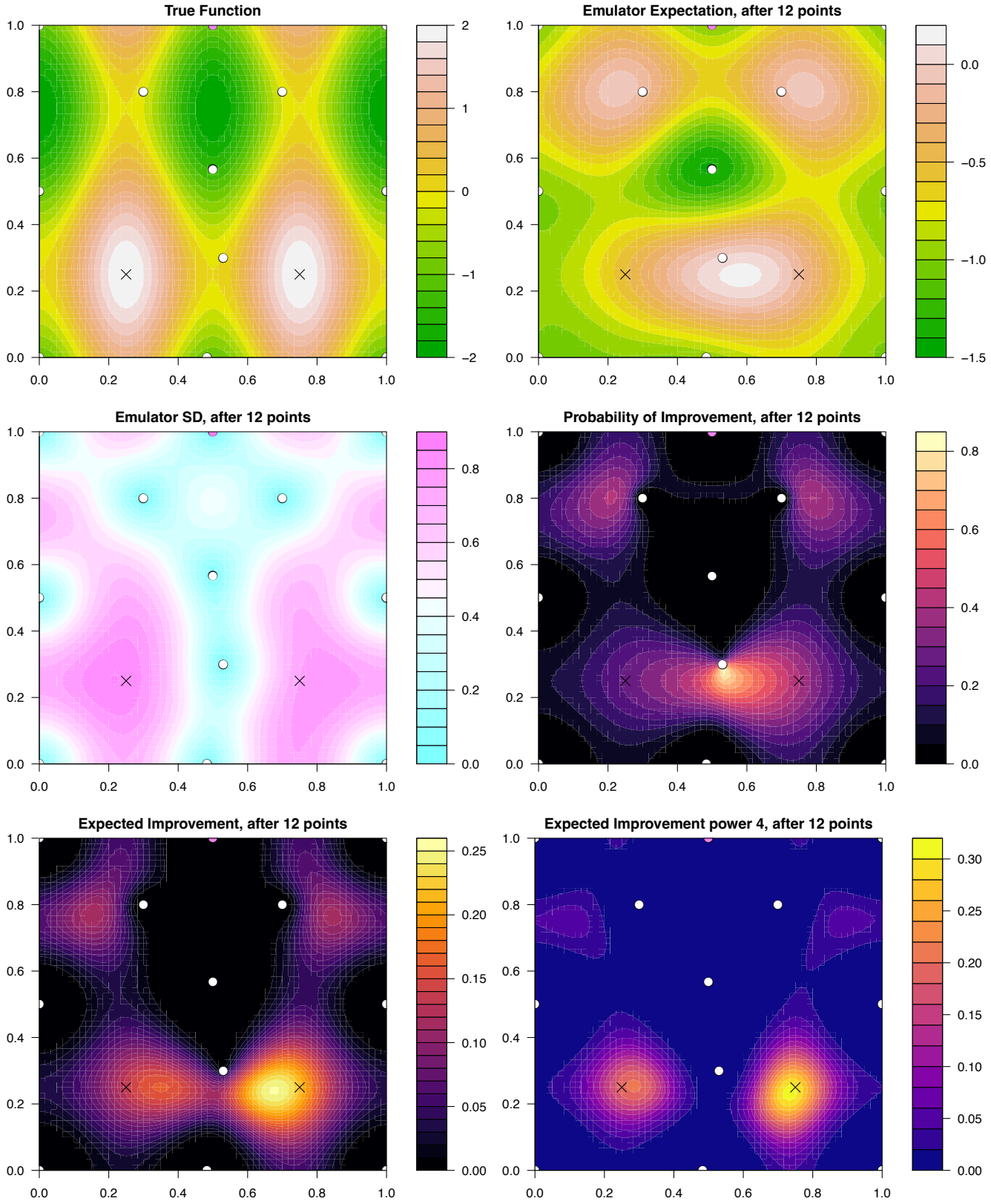


Figure 88: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

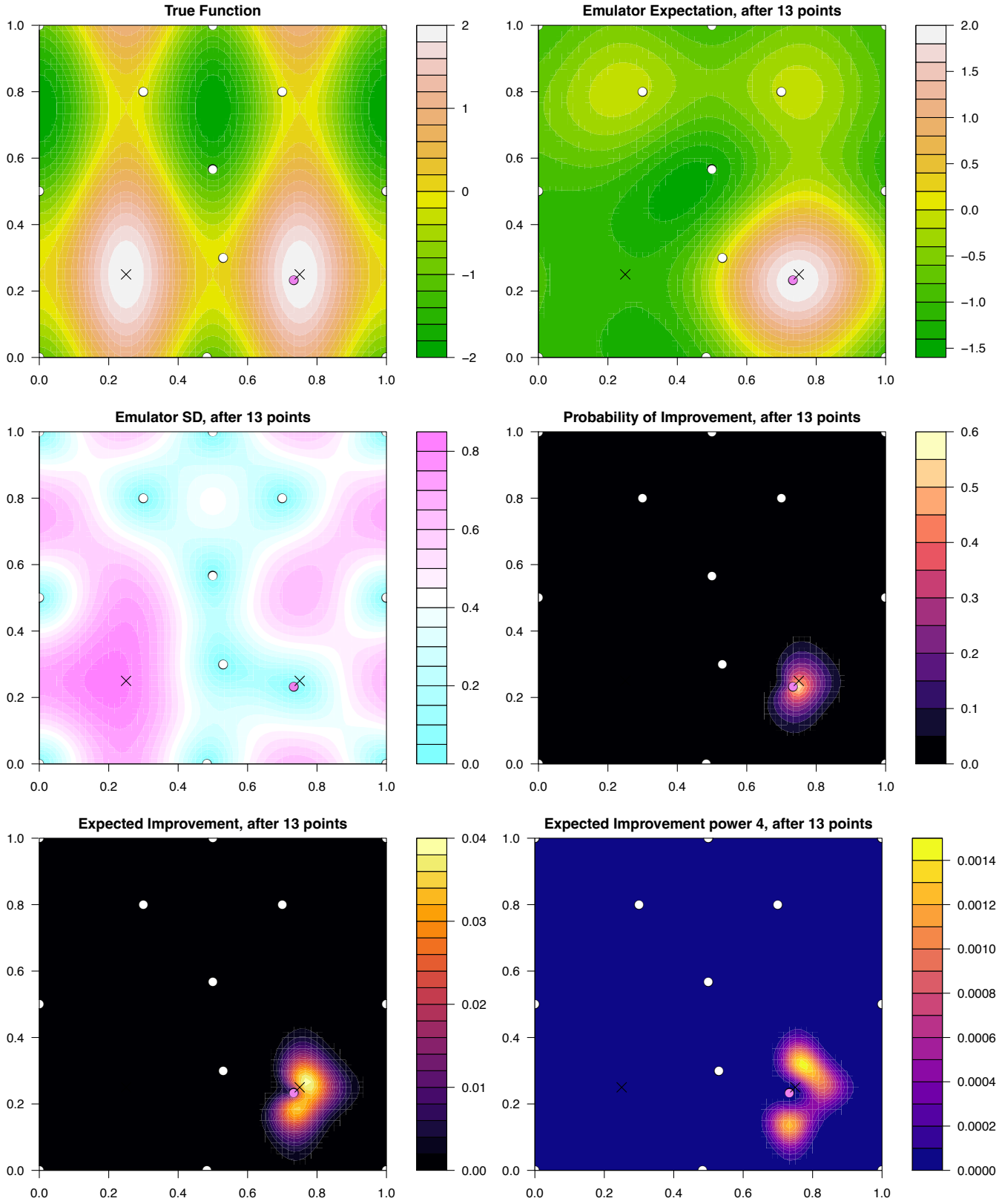


Figure 89: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

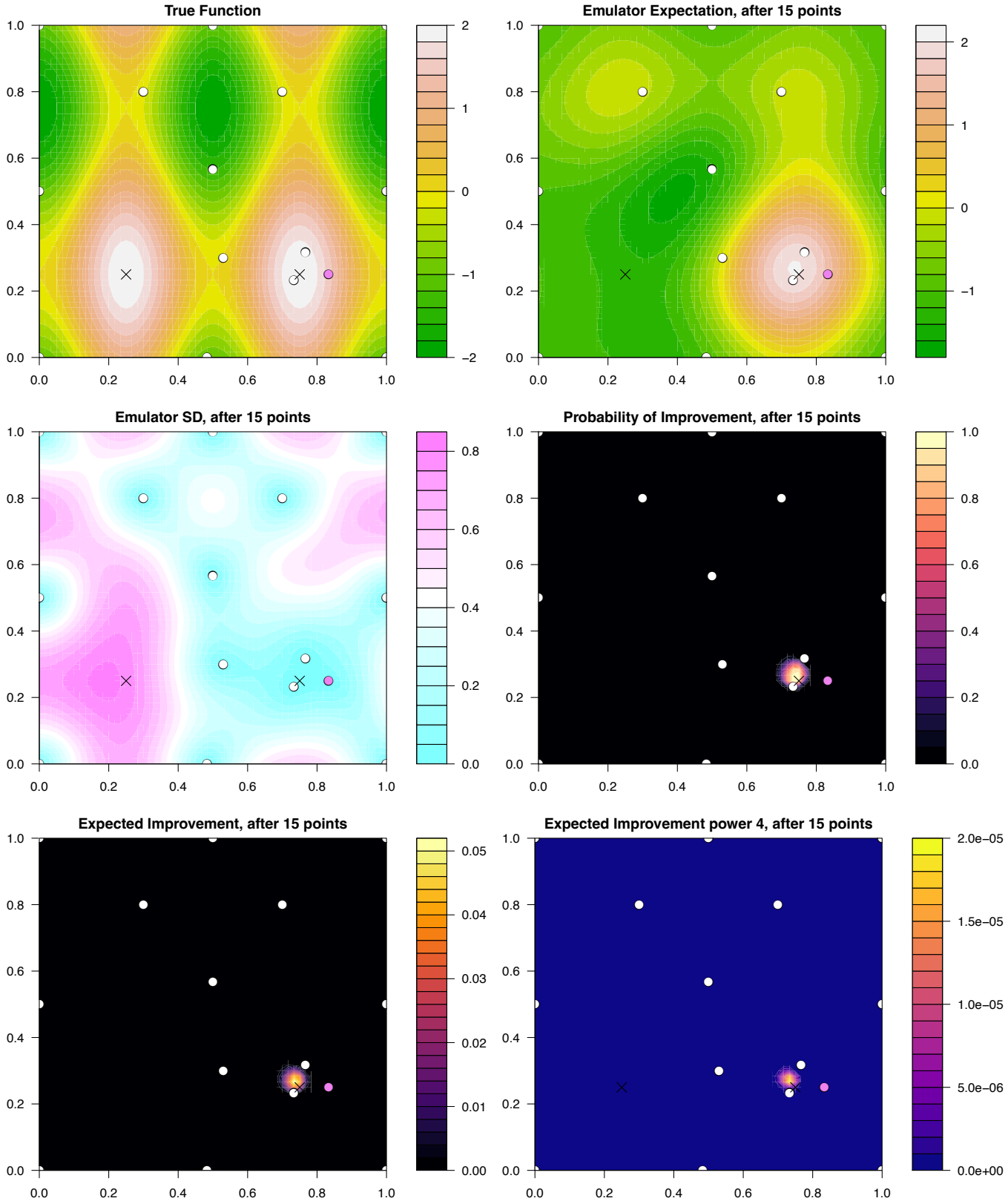


Figure 90: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

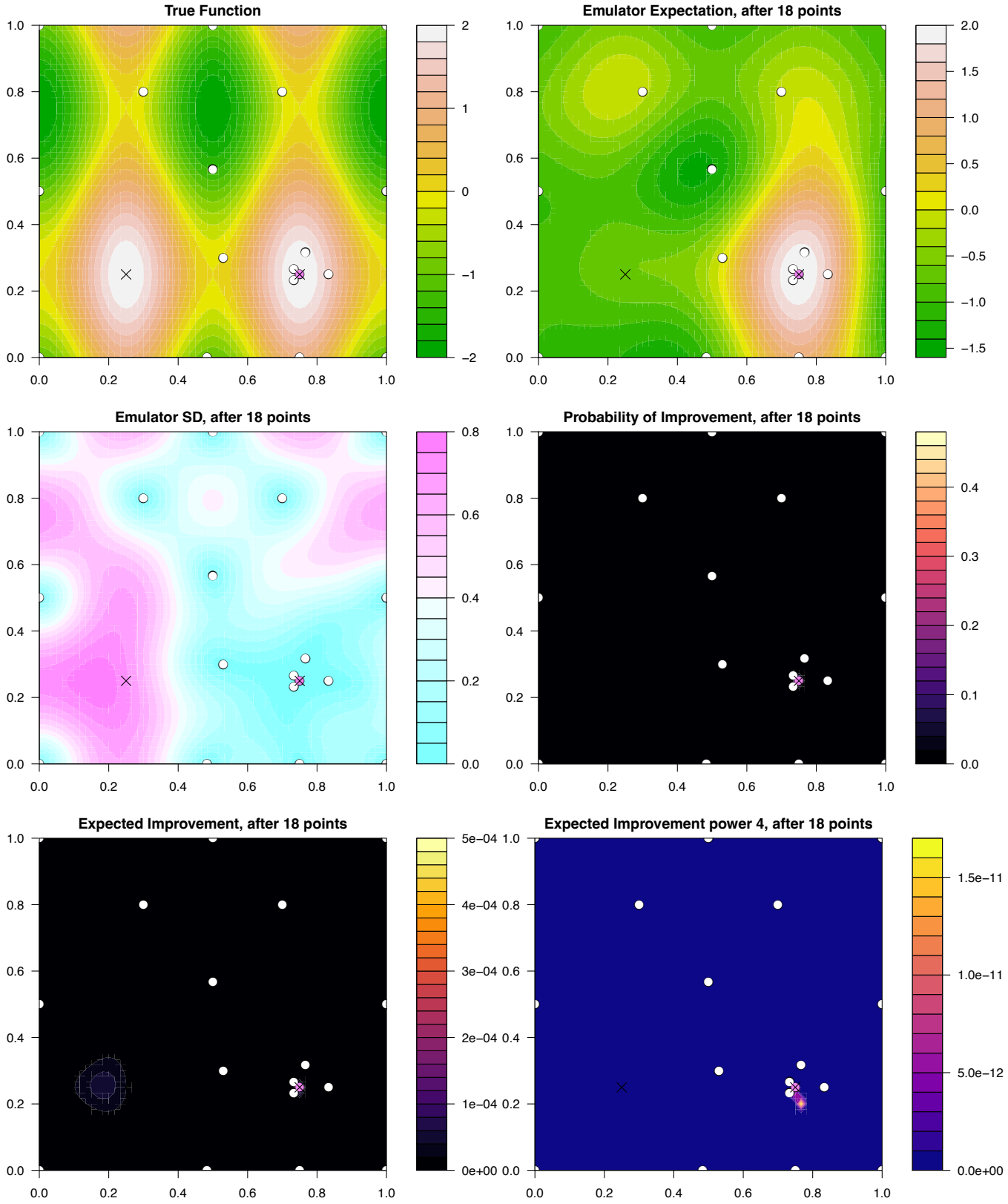


Figure 91: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

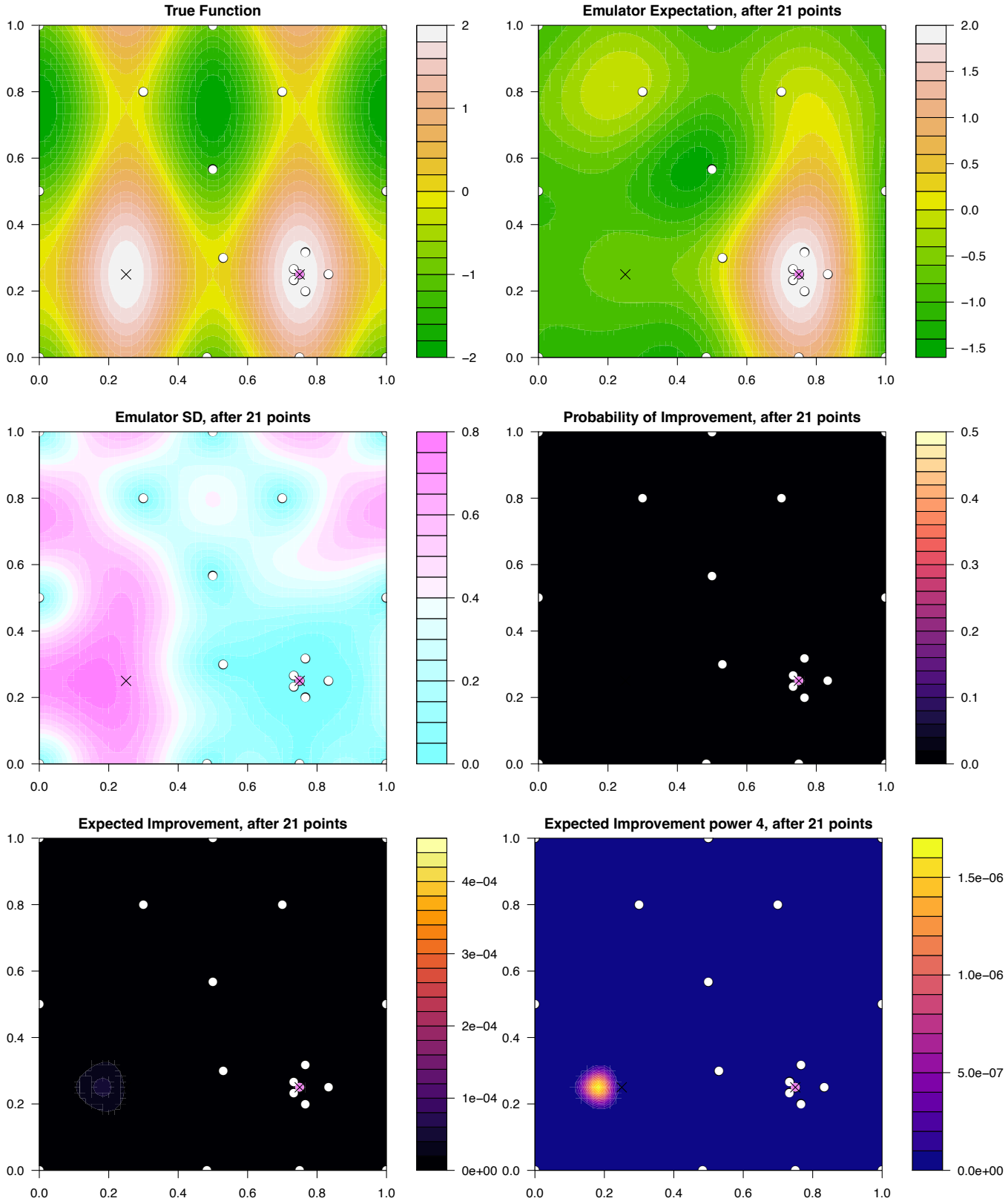


Figure 92: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

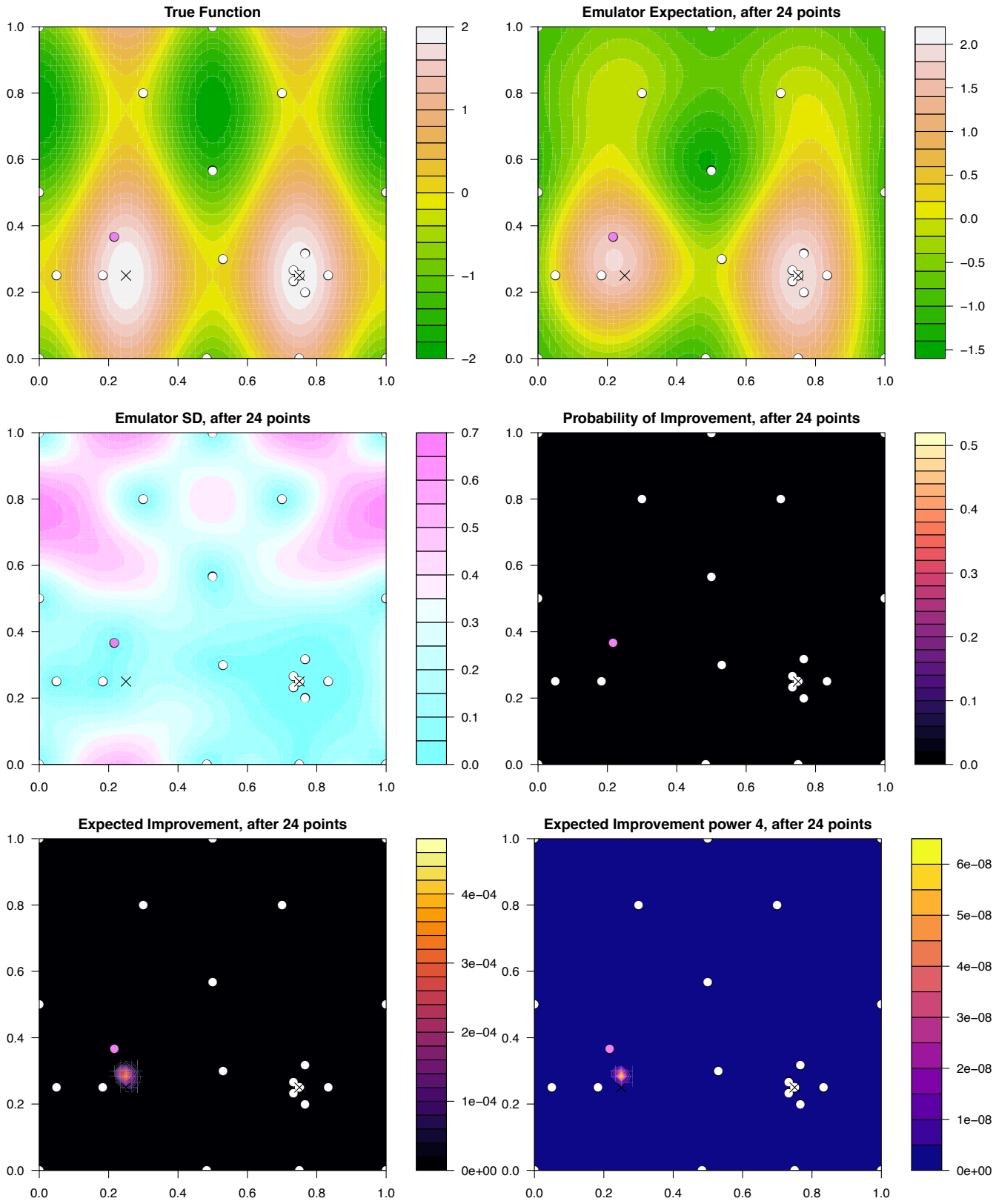


Figure 93: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

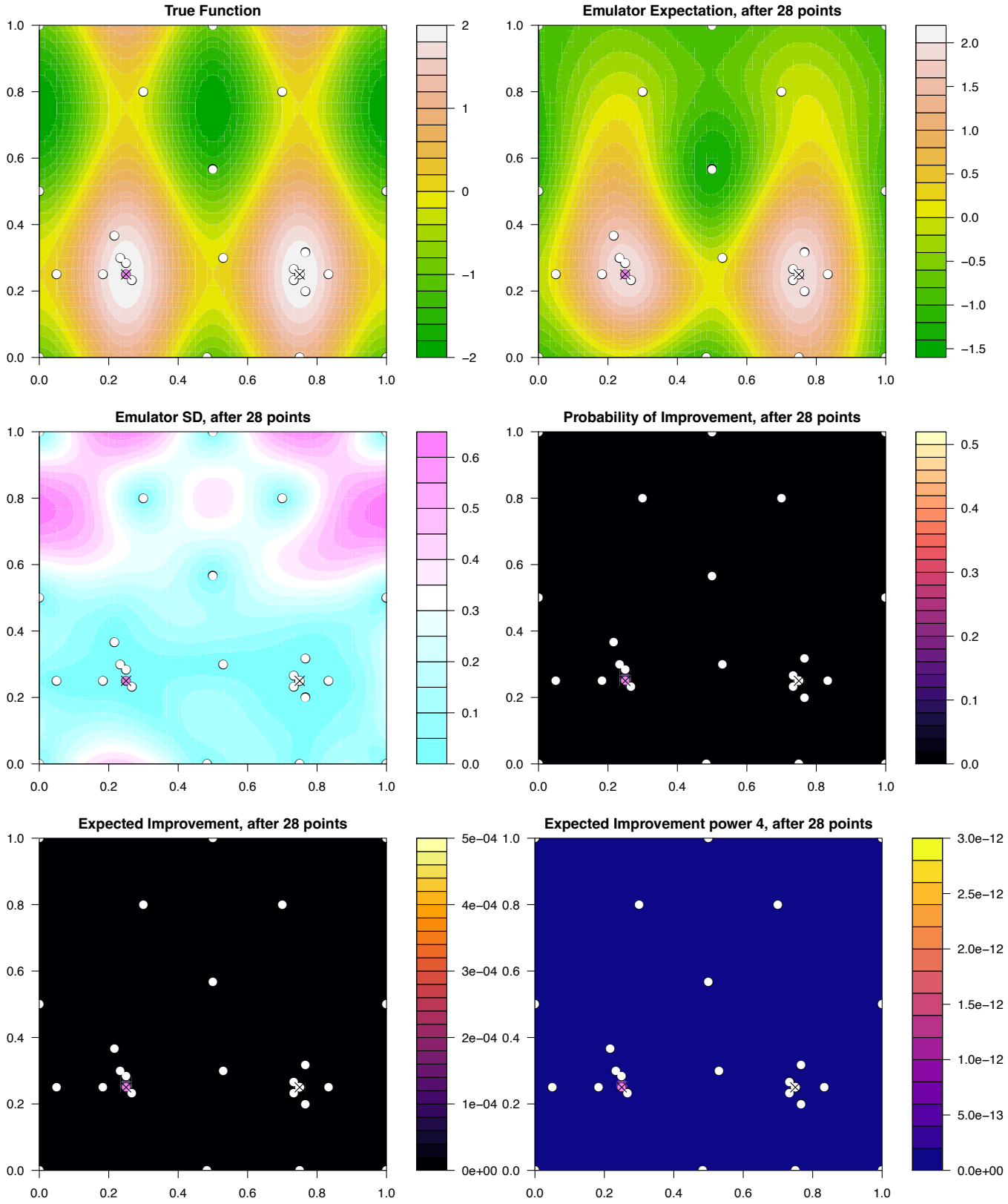


Figure 94: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

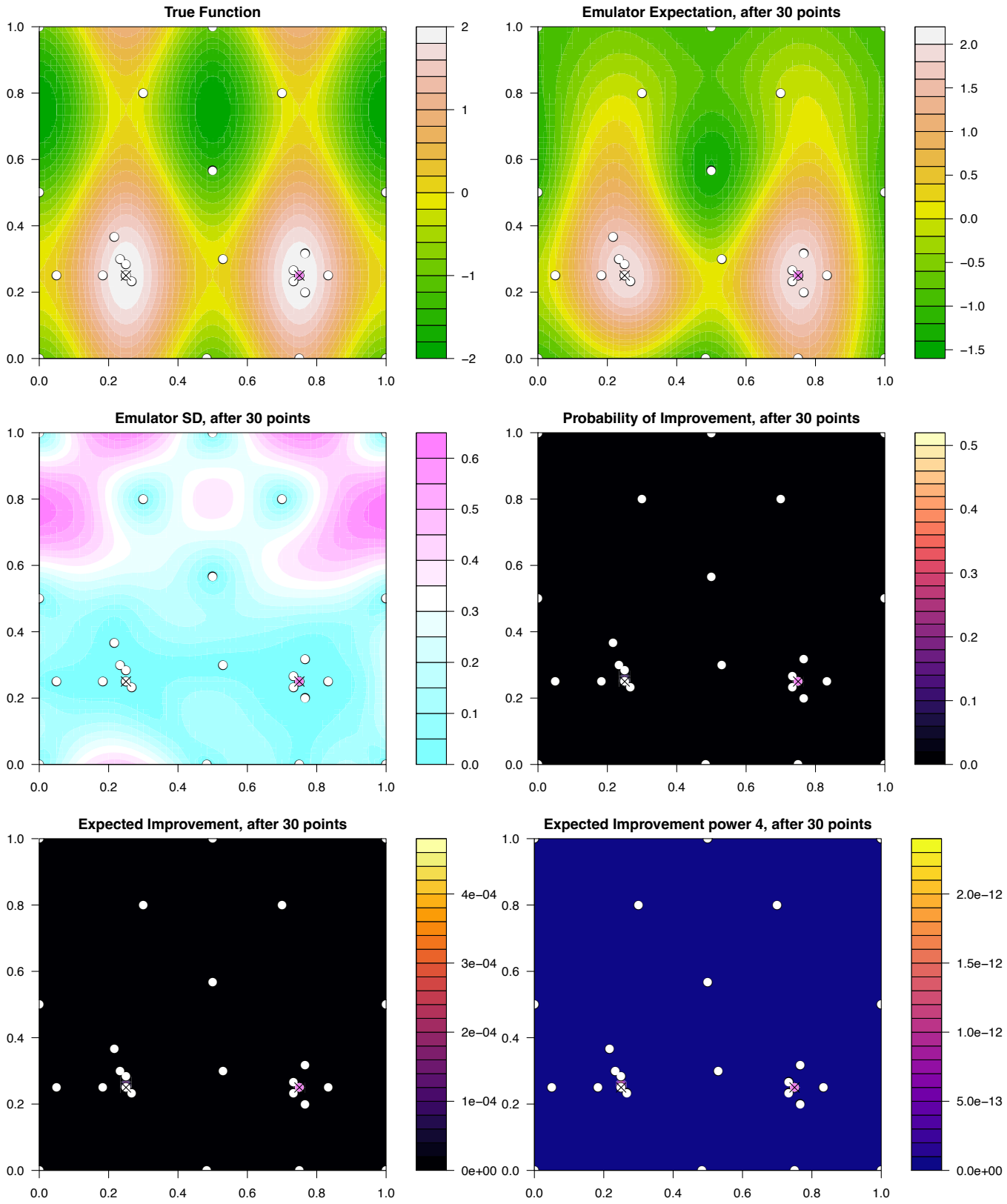


Figure 95: **Optimising using the $EI^4(x)$ criteria**: the 2-dimensional emulator of $f(x)$ discussed in section 7.2. Top left panel: the true function $f(x)$ over the 2-dimensional input space $\mathcal{X}$, with the white points showing the evaluated runs. The two maxima are shown as black crosses. Top right panel: the emulator expectation or mean $\mu_D(x)$. Middle left panel: the emulator standard deviation $\sigma_D(x)$. Middle right panel: the $PI(x)$ criteria, bottom left: the $EI(x)$ criteria and the bottom right the $EI_4(x)$ criteria (which is discussed and used in section 7.4). The most recent run is shown as the pink point, and total run number is given in the plot titles.

7.5 Sequential Decision Making for Optimisation

- While the above criteria are useful, and e.g $EI(x)$ is widely used, they are fundamentally flawed, in that they are all *myopic* criteria.
- This just means that they judge where to place the next run, but have no consideration of how many more runs we can perform.
- For example, the reason we like some exploration runs is not because we think they themselves will necessarily give very high values of $f(x)$, but that they will be very informative for the emulator and hence allow us to find even higher runs in the future.
- We know myopic strategies must be sub-optimal from sequential Decision Theory, which says that such approaches are limited (the full solution is to use backwards induction, however this is utterly intractable here!).
- What we can say is that a successful strategy will acknowledge how many runs we have left: e.g. if we only have computational resources for one more then using $EI(x)$ or its variants might be a good idea. However if we have many runs still to do, then deliberately choosing more exploratory runs at first, then chasing maxima in later runs, will most likely yield better results. This is somewhat achieved by the Implausibility/HM approach to optimisation that we performed in Computer Practical 4.
- How to design such sequential sets of runs more effectively is an active area of research within UQ.

8 Conclusion

We have finally reached the end of the Uncertainty Quantification IV course. I hope you have enjoyed the ideas and techniques that we have covered, and I would like to thank you for your hard work and patience throughout this term.

If you have any questions at any point about course content, problem sheet exercises, computer practicals or the mini-project do not hesitate to email me at: i.r.vernon@durham.ac.uk or drop by my office at any time.