HGF Toolbox Manual

Contents

1 Introduction

The HGF toolbox provides generic methods for fitting time series models using Bayesian inference.

It is built around the Hierarchical Gaussian Filter (HGF). It may therefore not contain your favorite time series model. However, the toolbox's modular nature should make it easy to add new models (see Section 6 below).

The HGF was introduced in Mathys et al. (2011) and elaborated in Mathys et al. (2014). Whenever you make use of one of the various models based on the HGF, please cite these papers.

As shown in Figure 1, the toolbox assumes a framework where an agent in the broadest sense (e.g., a human being, an animal, a machine, the stock market, etc.) receives a time series of inputs to which it reacts by emitting a time series of responses. In particular, this process is modeled by the combination of a perceptual (sc. state space) model and an observation (sc. decision or response) model. The perceptual model is the time series model on which the agent bases its responses; the response model describes how the agent makes decisions based on its perceptual inference.

Note that what we refer to here as the observation model describes a "second-order" observation in the sense that the perceptual model already contains a ("first-order") observation part that describes how perceptual states relate to inputs. This implements the "observing the observer" framework described in Daunizeau et al. (2010a).

2 Usage

There are two main ways to use the HGF toolbox. The first is to fit various combinations of perceptual and observation models to observed responses (Figure 2).

The second main usage is to simulate the trajectories of beliefs about external states, and responses based on these beliefs (Figure 3).

In simpler cases (e.g., when simply filtering inputs), only the evolution of the perceptual inference is of interest and the the specification of a response model is unnecessary (Figure 4).

3 Installation

Move the main folder with all its contents to a location of your choice and add it to your Matlab path.

4 Documentation and tutorial demo

Documentation is contained in this manual and throughout the code. To find out what a particular file does, consult the comments at the head of its source code.

A good way to get started with the toolbox is to do the tutorial demo by calling

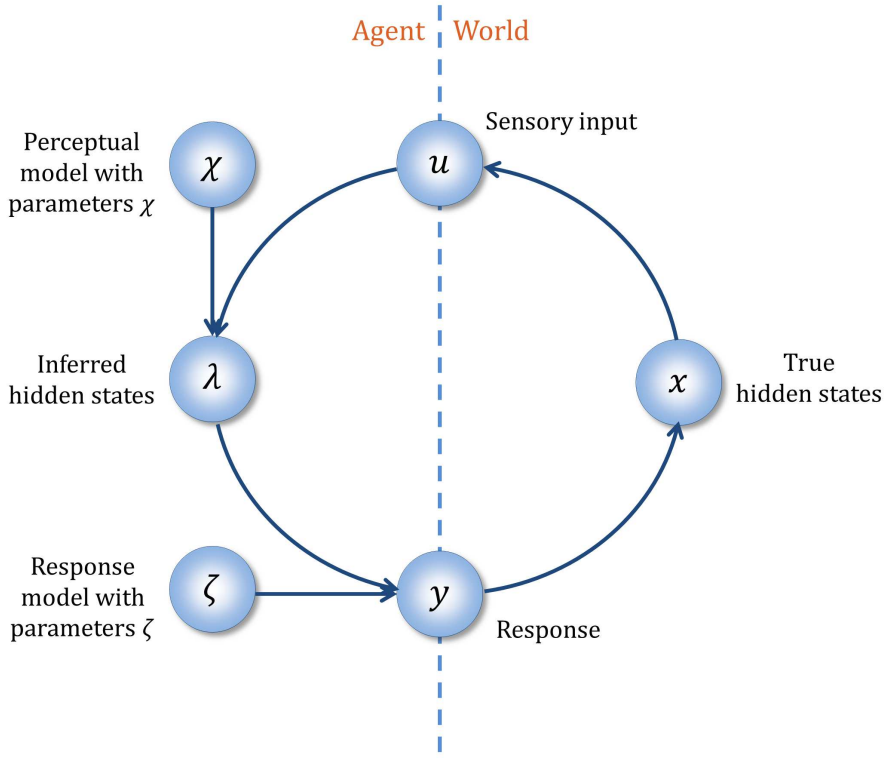

Figure 1: General framework. *An agent is connect to the external world by its sensory input u and by the actions y it takes in response. Inputs are used to infer hidden states of the world, beliefs λ about which are encoded internally. Inference rests on a perceptual model parameterized by χ. Actions depend on beliefs λ and are described by a response model parameterized by ζ.*

```
>> tapas_hgf_demo()
```

This walks you through the main ways to use the toolbox.

5 Main functions

Each of the two usages (cf. Section 2) has its main function. The function

```
tapas_fitModel(...)
```

fits models to observed responses, while the function

```
tapas_simModel(...)
```

simulates the trajectories of perceptual states and responses. These two main functions will be explained in turn in what follows.

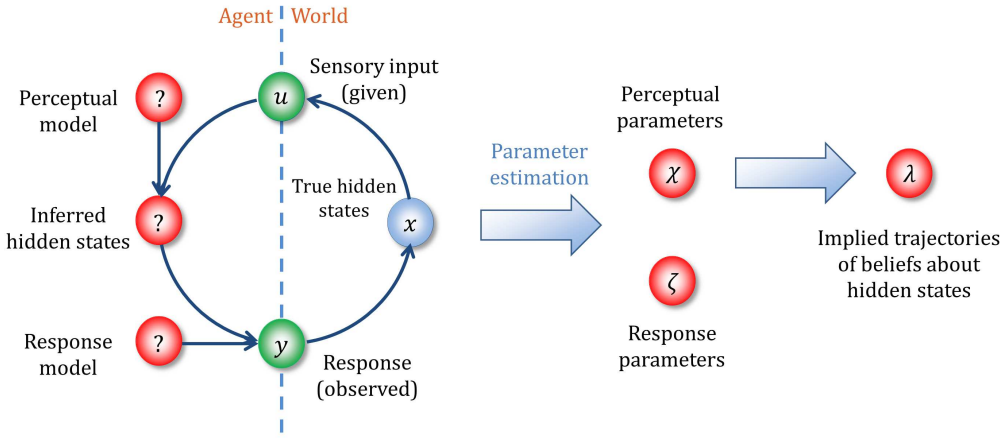

Figure 2: Parameter estimation. *When both inputs and responses are known, the parameters of the perceptual and the response model can be estimated, and the resulting estimate of the perceptual parameters implies trajectories of beliefs about hidden external states. Parameter estimation is performed by the function* `tapas_fitModel(...)`.

5.1 tapas_fitModel(...)

5.1.1 Background

In order to fit a model, you have to make three choices:

1) A perceptual model,

2) A response model, and

3) An optimization algorithm.

The perceptual model can for example be a Bayesian generative model of the states of an agent's environment (like the HGF) or a reinforcement learning algorithm (like Rescorla-Wagner). It describes the states or values that probabilistically determine observed responses.

The response model describes how the states or values of the perceptual model map onto responses. Examples are the softmax decision rule or the closely related unit-square sigmoid decision model.

The optimization algorithm is used to determine the maximum-a-posteriori (MAP) estimates of the parameters of both the perceptual and decision models. Its objective function is the unnormalized log-posterior of all perceptual and response parameters, given the data and the perceptual and response models. This corresponds to the log-joint

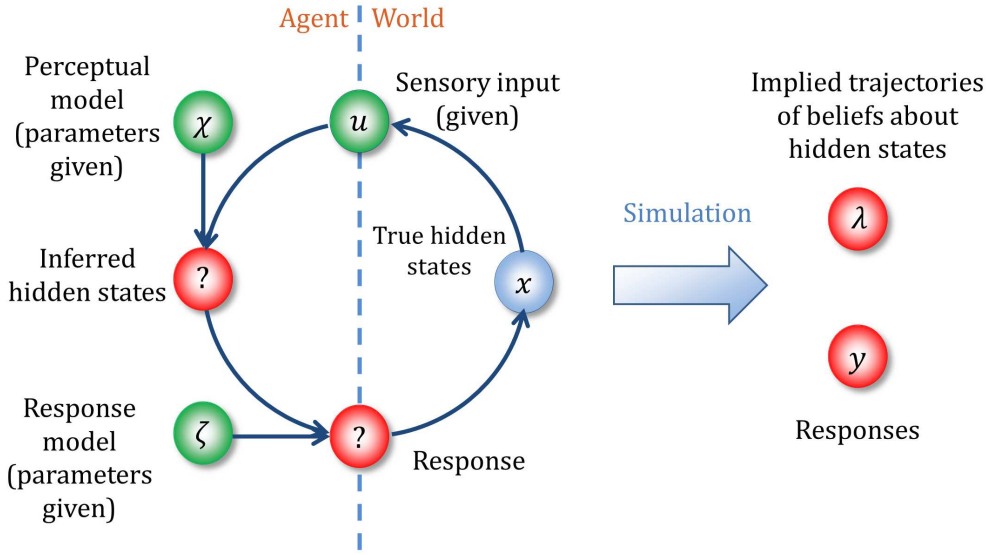

Figure 3: Simulation of belief trajectories and responses. *When the parameters of the perceptual and the response model are given in addition to the inputs, belief trajectories and responses can be simulated. This is performed by the function* `tapas_simModel(...)`.

of data and parameters, given the models.

Perceptual and response models have to be chosen so that they are compatible, while the choice of optimization algorithm is independent.

5.1.2 Configuration

Once you have made your choice, go to the relevant configuration files (e.g., `tapas_hgf_binary_config.m`), read the model- and algorithm-specific information there, and configure accordingly.

Usage then is:

```
>> est = tapas_fitModel(responses, inputs, <prc_model>,
                                    <obs_model>, <opt_algo>);
```

where the last three arguments are strings containing the names of the corresponding configuration files (without the extension `.m`).

These last three arguments are optional. If they are omitted the defaults configured in

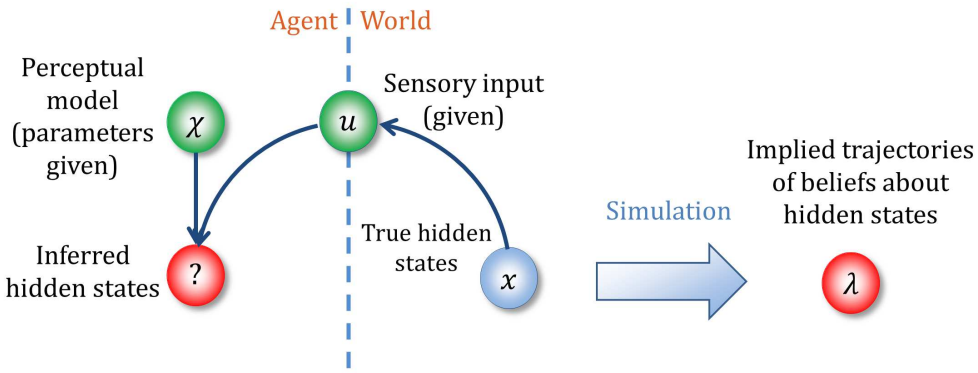

Figure 4: Simulation of belief trajectories only. *The function* `tapas_fitModel(...)` *can be called without specifying a response model.*

`tapas_fitModel.m` are used.

5.1.3 Using placeholder values

For inputs that lie on a continuous scale, the configuration files of some perceptual models accept placeholders that are replaced by values derived from the inputs at runtime (see the documentation in the relevant configuration files). This makes it easy to automatically set, for example, the prior mean of the main quantity of interest the value of the first input.

5.1.4 Usage

```
>> est = tapas_fitModel(responses, inputs);
```

5.1.5 Input arguments

`responses`

Array of binary responses (column vector(s))

`inputs`

Array of inputs (column vector(s))

Code irregular (missed, etc.) responses as `NaN`. Such responses will be ignored. However, the trial as such will not be ignored and filtering will take place based on the input.

To ignore a trial, code the input as `NaN`. In this case, filtering is suspended for this trial and all representations (i.e., inferences on hidden states) will remain constant.

Note that an input is often a composite event, for example a cue-stimulus contingency. If the agent you are modeling is learning such contingencies, inputs have to be coded in contingency space (e.g., blue cue → reward as well as green cue → no reward is coded as 1 while blue cue → no reward as well as green cue → reward is coded as 0). The same applies to responses.

If needed for a specific application, responses and inputs can be matrices with further columns. The coding of irregular and ignored trials described above then applies to their first column.

5.1.6 Output

`est.u`

Input to agent (i.e., the inputs array from the arguments)

`est.y`

Observed responses (i.e., the responses array from the arguments)

`est.irr`

Index numbers of irregular trials

`est.ign`

Index numbers of ignored trials

`est.c_prc`

Configuration settings for the chosen perceptual model (see the configuration file of the model in question for details)

`est.c_obs`

Configuration settings for the chosen observation model (see the configuration file of the model in question for details)

`est.c_opt`

Configuration settings for the chosen optimization algorithm (see the configuration file of the algorithm in question for details)

`est.optim`

A place where information on the optimization results is stored (e.g., measures of model quality like LME, AIC, BIC, and posterior parameter correlation)

`est.p_prc`

Maximum-a-posteriori estimates of perceptual parameters (see the configuration file of your perceptual model for details)

`est.p_obs`

Maximum-a-posteriori estimates of observation parameters (see the configuration file of your observation model for details)

`est.traj`

Trajectories of the environmental states tracked by the perceptual model (see the configuration file of that model for details)

5.1.7 New datasets

When analyzing a new dataset, take your inputs and use `tapas_bayes_optimal_config` (or `tapas_bayes_optimal_binary_config` for binary inputs, or similar depending on your perceptual model) as your observation model. This determines the Bayes optimal perceptual parameter values (i.e., the parameter values that result in the agent being least surprised overall at the inputs it receives). You can then use the optimal parameter values as your new prior means.

5.1.8 Plotting of results

To plot the trajectories of the inferred perceptual states (as implied by the estimated parameters), there is a function `tapas_<modelname>_plotTraj(...)` for each perceptual model. This takes the structure returned by `tapas_fitModel(...)` as its only argument.

Additionally, the function `tapas_fit_plotCorr(...)` plots the posterior correlation of the estimated parameters. It takes the structure returned by `tapas_fitModel(...)` as its only argument. Note that this function only works if the optimization algorithm makes the posterior correlation available in `est.optim.Corr`.

5.1.9 Example

```
>> est = tapas_fitModel(responses, inputs);
>> tapas_hgf_binary_plotTraj(est)
>> tapas_fit_plotCorr(est)
```

5.1.10 Bayesian parameter averaging

It is often useful to average parameters from several estimations, for instance to compare groups of subjects. This can be achieved by using the function `tapas_bayesian_parameter_average(...)` which takes into account the covariance structure between the parameters and weights individual estimates according to their precision.

Bayesian parameter averaging only works for estimates that are based on the same priors and should only be used with care for estimates based on different inputs.

5.1.11 Model comparison

For each model estimation, the toolbox calculates three measures of model goodness that can be used for model comparison. The first of these is the log-model evidence (LME), calculated as the negative variational free energy under the Laplace assumption. In addition to the LME, AIC and BIC are calculated, which in turn are approximations to the negative variational free energy. This makes the LME the measure of choice for model comparison, while AIC and BIC are only provided because some audiences are more familiar with them.

LMEs can be used to calculate Bayes factors by exponentiating the difference in LME between two models applied to the same dataset. For example, an LME difference of 3 implies a Bayes factor of about 20.

For a fixed-effects analysis with several datasets (e.g., from different subjects), add up the LMEs for the different datasets and compare the LME sums.

For a random-effects analysis, use the function `spm_BMS(...)` from the SPM software package (http://www.fil.ion.ucl.ac.uk/spm/) and see Stephan et al. (2009) for the theoretical background.

5.2 tapas_simModel(...)

5.2.1 Background

In order to simulate perceptual states (and, optionally, responses) in this framework, one has to choose a perceptual model (and, optionally, an observation model to simulate responses).

The perceptual model can for example be a Bayesian generative model of the states of an agent's environment (like the HGF) or a reinforcement learning algorithm (like Rescorla-Wagner). It describes the states or values that probabilistically determine observed responses.

The observation model describes how the states or values of the perceptual model map onto responses. Examples are the softmax decision rule or the closely related unit-square sigmoid decision model.

5.2.2 Usage

```
>> sim = tapas_simModel(inputs, prc_model, prc_pvec,
                                    obs_model, obs_pvec);
```

5.2.3 Input arguments

`inputs`

Array of inputs (column vector(s))

To ignore the input of a trial, code the input as NaN. In this case, filtering is suspended for this trial and all representations (i.e., inferences on hidden states) will remain constant.

Note that an input is often a composite event, for example a cue-stimulus contingency. If the agent you are modeling is learning such contingencies, inputs have to be coded in contingency space (e.g., blue cue → reward as well as green cue → no reward is coded as 1 while blue cue → no reward as well as green cue → reward is coded as 0). The same applies to responses.

If needed for a specific application, inputs can be a matrix with further columns. The coding of ignored trials described above then applies to its first column.

`prc_model`

The perceptual model (e.g., `tapas_hgf` or `tapas_hgf_binary`)

`prc_pvec`

Row vector of perceptual model parameter values (see the corresponding model's configuration file).

`obs_model`

The observation model (e.g., `tapas_gaussian_obs` or `tapas_softmax_binary`)

`obs_pvec`

Row vector of observation model parameter values (see the corresponding model's configuration file).

The last two input arguments are optional. If they are missing, no responses will be simulated.

To learn more about the various perceptual and observation models, refer to the comments in their configuration files (e.g., for `tapas_hgf_binary` to `tapas_hgf_binary_config.m`).

5.2.4 Output

`sim.u`

Input to agent (i.e., the inputs array from the arguments)

`sim.ign`

Index numbers of ignored trials

`sim.c_sim`

Information on the models used in the simulation

`sim.p_prc`

The perceptual parameters as given in `pvec_prc` in the configuration file of the perceptual model.

`sim.p_obs`

The observation parameters as given in `pvec_obs` in the configuration file of the observation model.

`sim.traj`

Trajectories of the environmental states tracked by the perceptual model (see the configuration file of that model for details)

`sim.y`

Simulated responses

5.2.5 Plotting of results

To plot the trajectories of the perceptual states implied by the chosen parameter values, there is a function `tapas_<modelname>_plotTraj(...)` for each perceptual model. This takes the structure returned by `tapas_simModel(...)` as its only argument.

5.2.6 Example

```
>> sim = tapas_simModel(u, 'tapas_hgf_binary',
             [NaN 0 1 NaN 1 1 NaN 0 0 NaN 1 NaN -2.5 -6],
             'tapas_unitsq_sgm', 5);
>> tapas_hgf_binary_plotTraj(sim)
```

6 Adding models

Owing to its modular structure, the toolbox allows you to add perceptual and observation models of your choice. This requires the following files containing the corresponding functions that `tapas_fitModel(...)` and `tapas_simModel(...)` will expect to find (replace `<modelname>` by the name of your model):

`tapas_<modelname>.m`

Contains the model machinery

`tapas_<modelname>_config.m`

Contains the configuration settings (only for `tapas_fitModel(...)`)

`tapas_<modelname>_transp.m`

Transforms parameters from the space they are estimated in to their native space (only for `tapas_fitModel(...)` and `tapas_bayesian_parameter_average(...)`)

`tapas_<modelname>_namep.m`

Returns a structure of named parameters (only for `tapas_simModel(...)`)

Additionally, for observation models, `tapas_simModel(...)` expects to find a function that performs the simulation of responses:

`tapas_<modelname>_sim.m`

For details, look at the corresponding files of an existing model (e.g., `tapas_hgf_binary.m`, etc.) and use them as templates.

7 Adding optimization algorithms

To add a new optimization algorithm, provide the following functions that `tapas_fitModel(...)` will expect to find (replace `<algo>` by the name of your algorithm):

`tapas_<algo>`

Contains the machinery of the algorithm

`tapas_<algo>_config`

Contains the configuration settings

For details, see the corresponding files of an existing algorithm (e.g., `tapas_quasinewton_optim.m`) and use them as templates.

8 Selection of published studies using the toolbox

In addition to papers cited throughout, the references at the end of this manual contain a selection of published studies where the HGF Toolbox was used.

References

Daunizeau, J., den Ouden, H. E. M., Pessiglione, M., Kiebel, S. J., Friston, K. J. & Stephan, K. E. (2010a). Observing the observer (II): Deciding when to decide. *PLoS ONE*, **5**(12), e15 555.

Daunizeau, J., den Ouden, H. E. M., Pessiglione, M., Kiebel, S. J., Stephan, K. E. & Friston, K. J. (2010b). Observing the observer (i): Meta-bayesian models of learning and decision-making. *PLoS ONE*, **5**(12), e15 554.

Diaconescu, A. O., Mathys, C., Weber, L. A. E., Daunizeau, J., Kasper, L., Lomakina, E. I., Fehr, E. & Stephan, K. E. (2014). Inferring on the intentions of others by hierarchical bayesian learning. *PLoS Comput Biol*, **10**(9), e1003 810.

Hauser TU, Iannaccone R, Ball J & et al (2014). Role of the medial prefrontal cortex in impaired decision making in juvenile attention-deficit/hyperactivity disorder. *JAMA Psychiatry*.

Iglesias, S., Mathys, C., Brodersen, K., Kasper, L., Piccirelli, M., denOuden, H. & Stephan, K. (2013). Hierarchical prediction errors in midbrain and basal forebrain during sensory learning. *Neuron*, **80**(2), 519–530.

Mathys, C., Daunizeau, J., Friston, K. J. & Stephan, K. E. (2011). A bayesian foundation for individual learning under uncertainty. *Frontiers in Human Neuroscience*, **5**, 39.

Mathys, C. D., Lomakina, E. I., Daunizeau, J., Iglesias, S., Brodersen, K. H., Friston, K. J. & Stephan, K. E. (2014). Uncertainty in perception and the hierarchical gaussian filter. *Frontiers in Human Neuroscience*, **8**, 825.

Schwartenbeck, P., FitzGerald, T. H. B., Mathys, C., Dolan, R. & Friston, K. (2014). The dopaminergic midbrain encodes the expected certainty about desired outcomes. *Cerebral Cortex*.

Stephan, K. E., Penny, W. D., Daunizeau, J., Moran, R. J. & Friston, K. J. (2009). Bayesian model selection for group studies. *NeuroImage*, **46**(4), 1004–1017.

Vossel, S., Bauer, M., Mathys, C., Adams, R. A., Dolan, R. J., Stephan, K. E. & Friston, K. J. (2014a). Cholinergic stimulation enhances bayesian belief updating in the deployment of spatial attention. *The Journal of Neuroscience*, **34**(47), 15 735–15 742.

Vossel, S., Mathys, C., Daunizeau, J., Bauer, M., Driver, J., Friston, K. J. & Stephan, K. E. (2014b). Spatial attention, precision, and bayesian inference: A study of saccadic response speed. *Cerebral Cortex*, **24**(6), 1436–1450.