Optimal Learning: an Overview

Peter I. Frazier

Operations Research & Information Engineering, Cornell University

Thursday June 12, 2014
Guest Lecture, Operations Research 3 – Decision-making
Tsinghua University

Research supported by AFOSR and NSF

What is optimal learning?

- In many applications, we make decisions about which data to collect.
- In making these decisions we trade the **benefit** of information (the ability to make better decisions in the future) against its **cost** (money, time, or opportunity cost).
- Statistical learning is making predictions or decisions based on data.
- **Optimal learning is making decisions about which data to collect in an optimal way.**

Optimal learning overlaps with other fields

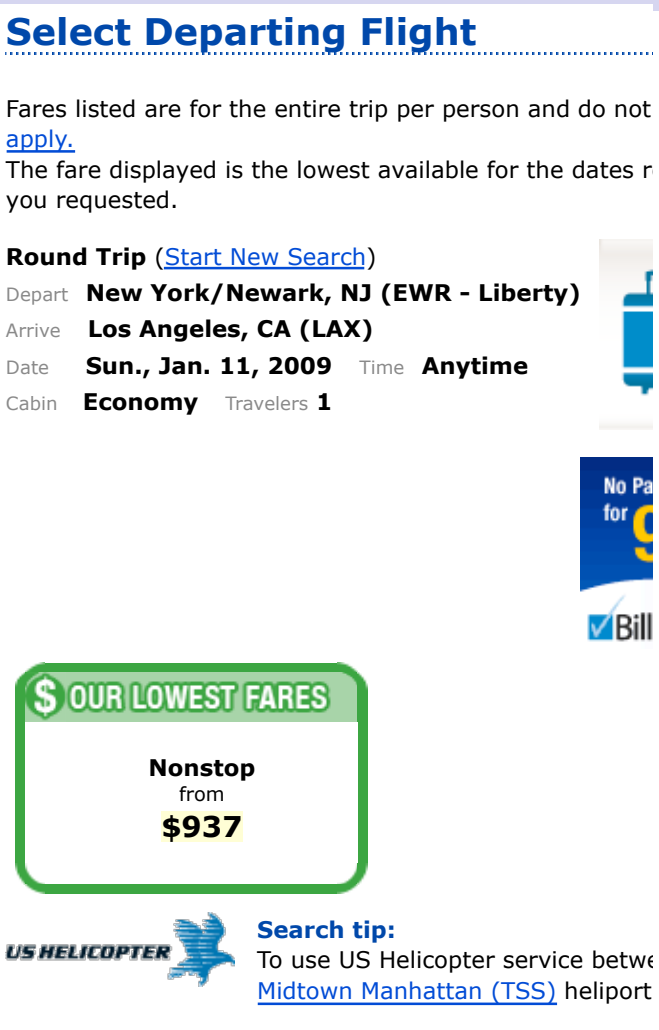
Optimal learning overlaps with these fields:

- Bayesian statistics, and machine learning.
- Decision-making under uncertainty, and dynamic programming.

Outline

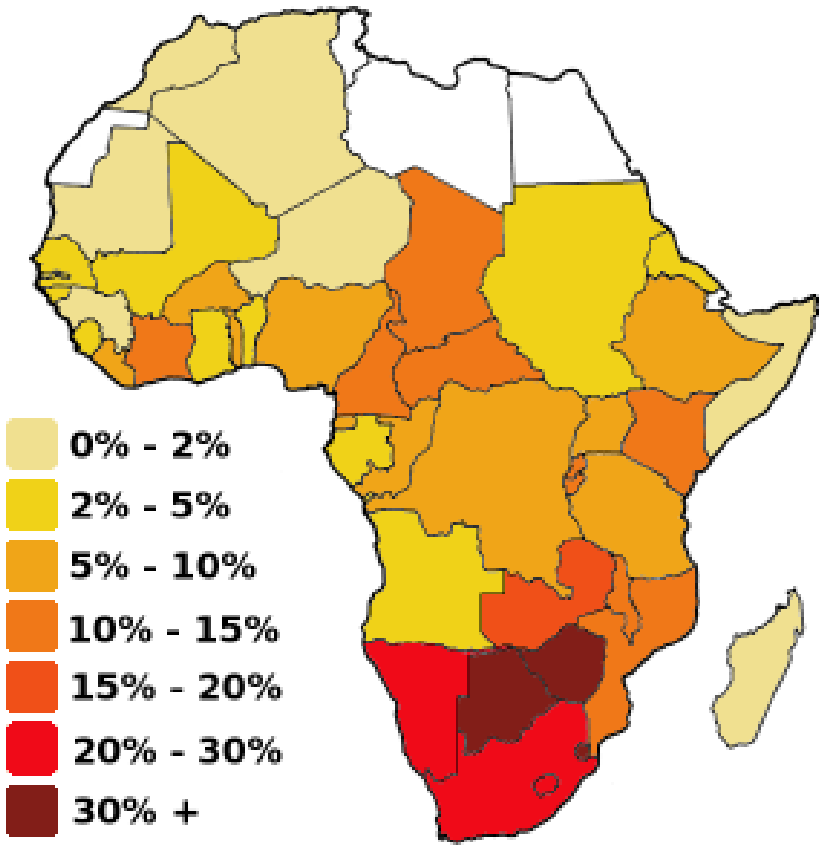
Dynamic Pricing

- Our goal is to price airline tickets to maximize revenue.
- We learn about demand for a flight as we sell tickets.
- The information collected depends on how we price each ticket: we only observe whether the price that the customer was willing to pay was above or below the offered price.
- Collecting more information now may provide the ability to improve revenues later.



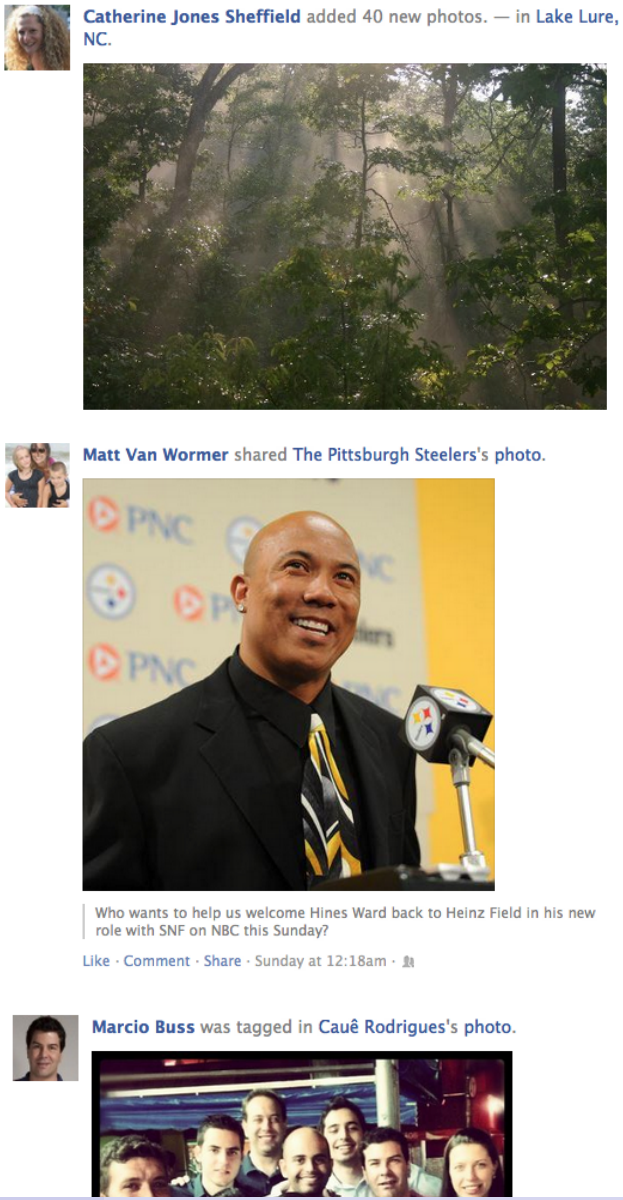
AIDS Treatment and Prevention

- We would like to treat and prevent AIDS in Africa.
- We are uncertain about the effectiveness of experimental treatments and untried prevention methods, but we can learn about them by using them in practice, or by conducting scientific studies.
- To which treatment and prevention methods should we allocate our investigative resources?
- How should we balance using those methods that appear to be most effective, with those untried methods that may be very good?

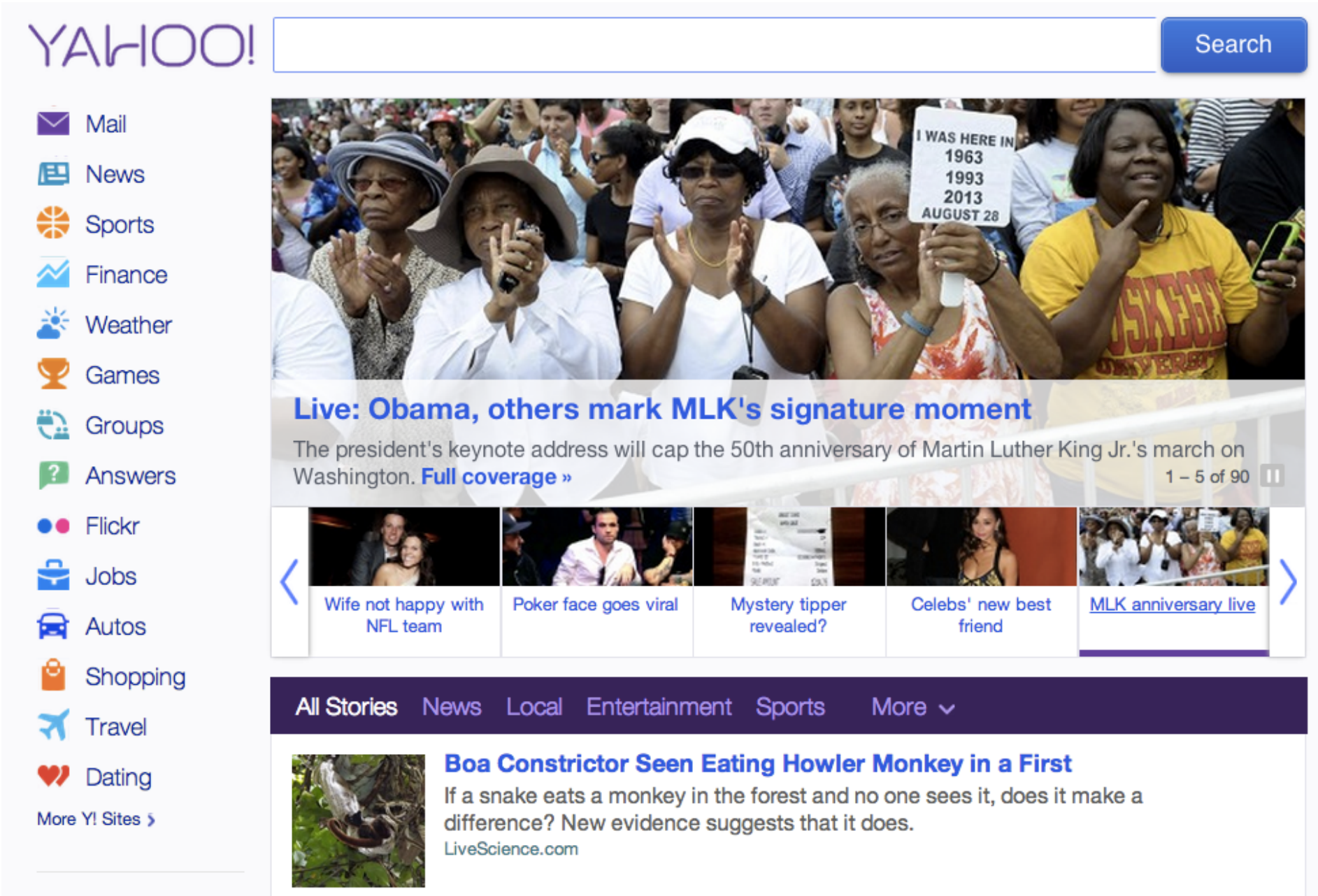


Exploration vs. Exploitation in News Feeds

- We would like to design an automatic document screening system that forwards documents (e.g., webpages) of interest to a human.
- The screening system earns a reward if the forwarded document is of interest, and pays a penalty if not.
- Even if the expected *immediate* reward of forwarding a particular document is negative, the system may still want to do so because human feedback may allow the system to improve future performance.



Adaptive Web Design (multi-armed bandits)



Product development (optimization of expensive functions)

- We have a product whose features we are selecting based on a sequence of focus groups.
- We have the time and budget for a fixed number of focus groups, through which we want to learn more about underlying consumer preferences for these features.
- After conducting these focus groups, we will choose a particular set of features with which to bring our product to market and receive a reward based on the resulting sales revenue and manufacturing and development costs.

Other examples

- Materials informatics / Designing novel materials
- Simulation optimization
- Optimization of long-running computer codes
- Clinical trials (sequential hypothesis testing)
- Inventory control with censored demand
- Quality control (changepoint detection)

Outline

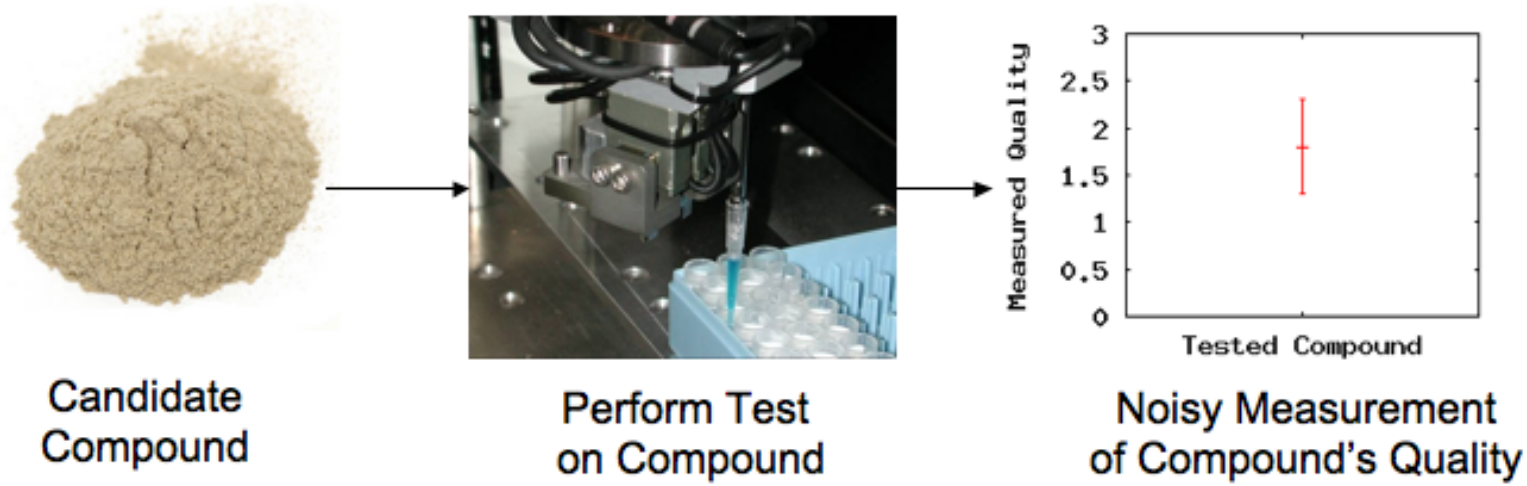
Outline

We consider an optimal learning problem called Bayesian Ranking & Selection

- We consider an optimal optimal learning problem called "Bayesian Ranking & Selection (R&S)" or "Bayesian Selection of the Best".
- In this problem, we wish to know which of a finite number of options is the best.
- To figure out the quality of an option, we can sample it (try it out).
- When we sample an option, we get a noisy observation of its quality.
- We can take a limited number of samples.
- We wish to allocate this sampling budget efficienly, so as to best support selecting the best.

Example: Drug Discovery

A pharmaceutical company has a library of millions of compounds that it would like to screen for potential cancer drugs. Robots will do the initial assay by performing a fixed test one or several times on some subset of the compounds.

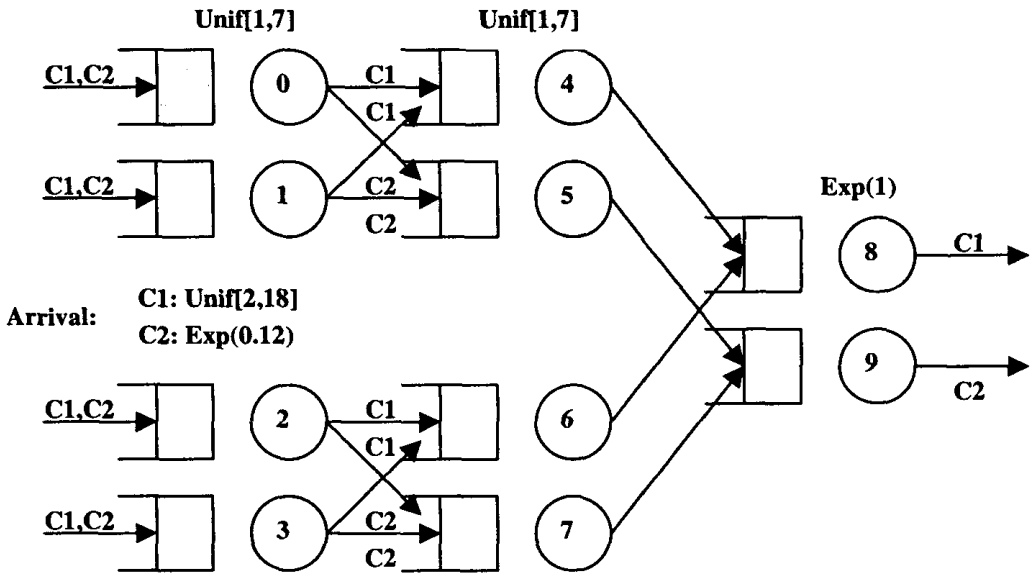


Sources: http://www.paa.co.uk/img/labauto/inst_highres/ssi/mini_dispenser.jpg,

http://www.kalyx.com/store/images/Images_SW/SW_201442-51.jpg

Example: Queuing Control

- We would like to choose a nurse/doctor staffing policy in a hospital to minimize expected patient waiting time.
- To figure out the patient waiting time under a particular staffing policy is, we can simulate it using a discrete event simulation.
- Each simulation takes about 1 minute.
- We want to choose the best among 100 possible staffing policies, using at most 24 hours of simulation effort.



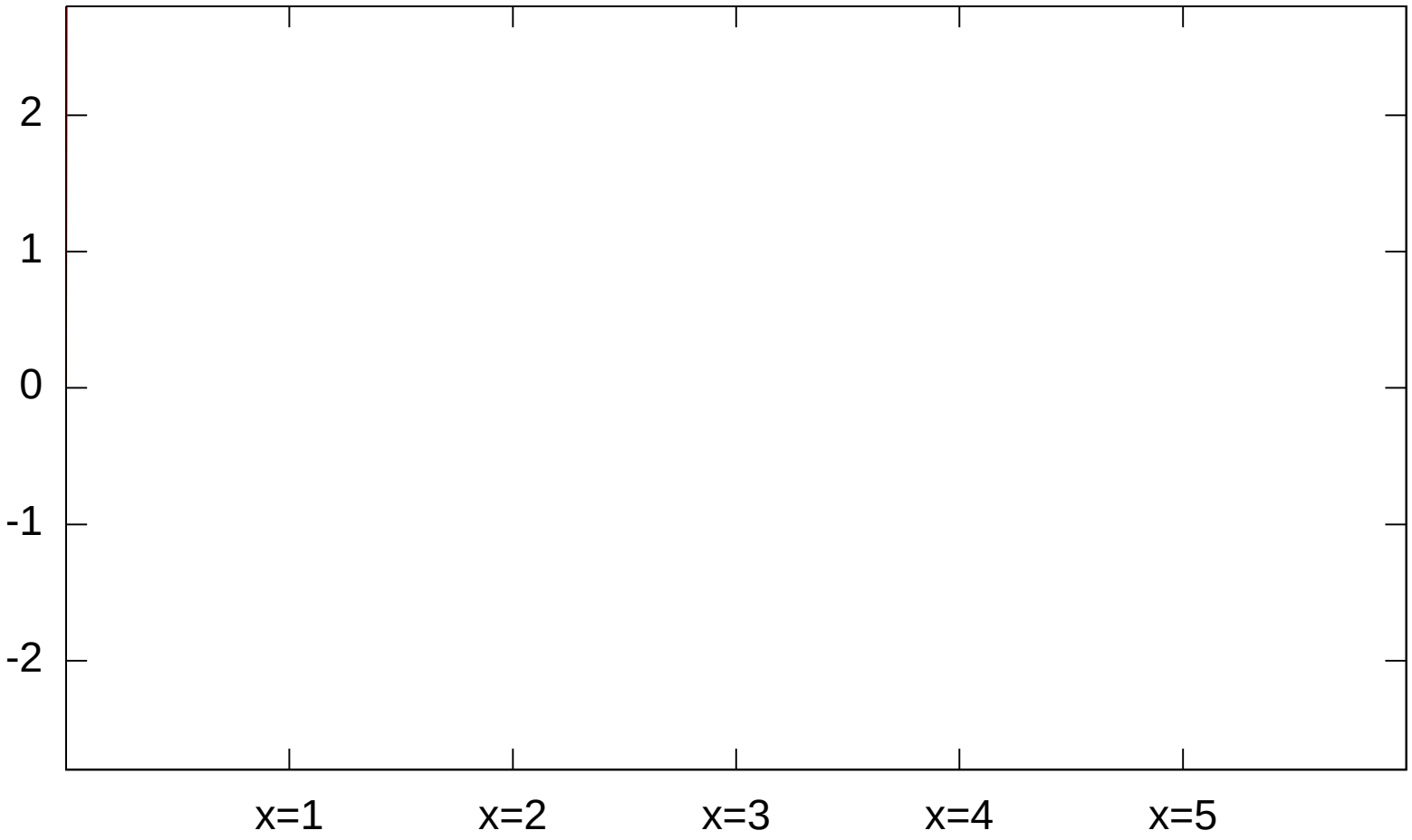
Mathematical Model

- We consider k alternative options.
- The underlying value of alternative x is $\theta_x \in \mathbb{R}$. We do not observe this, and must try to learn it through sampling. Let $\theta = (\theta_1, \ldots, \theta_k)$.
- At each time $n = 1, \ldots, N$, we choose an alternative to sample, $x_n \in \{1, \ldots, k\}$.
- We observe a sample,

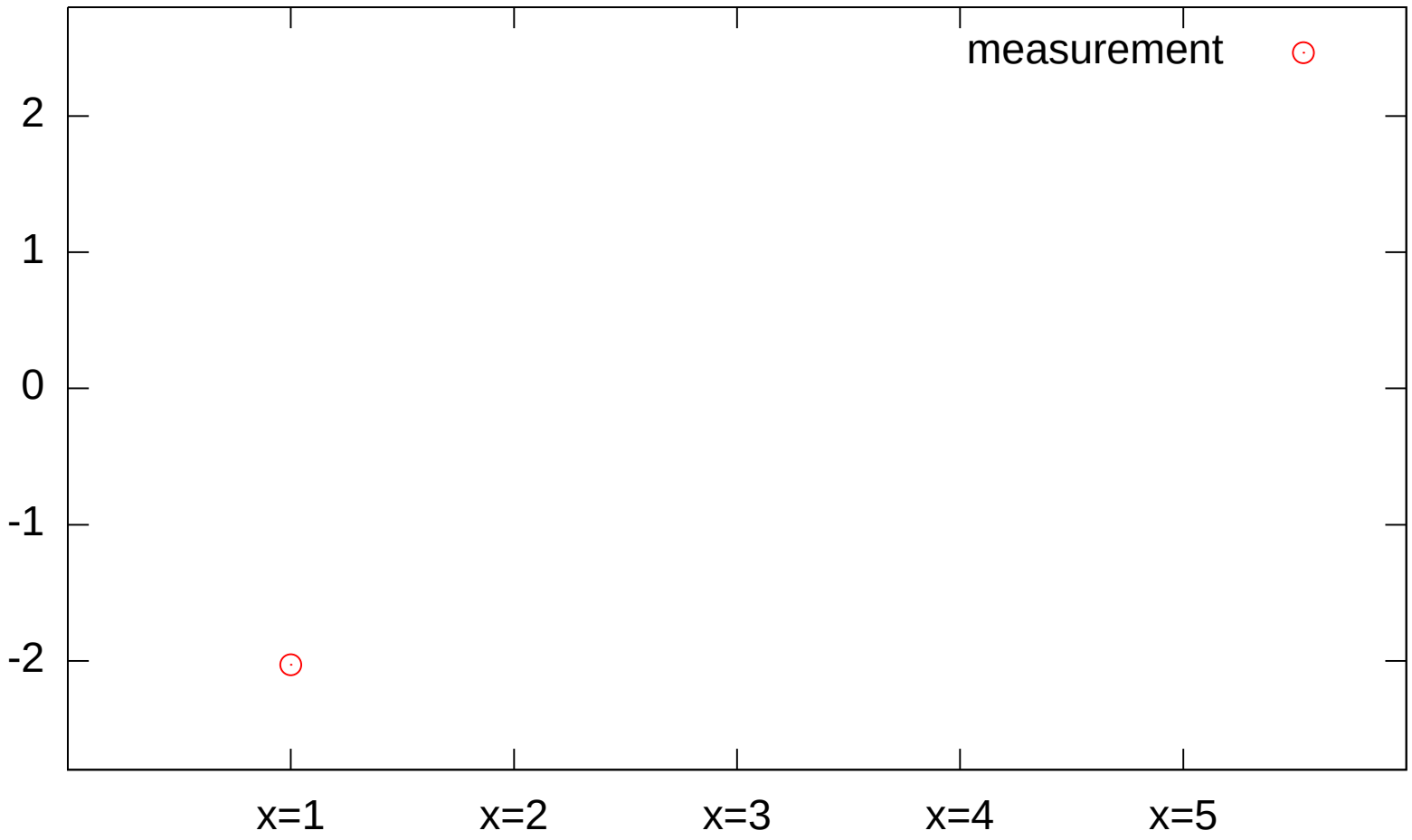
$$y_n \mid x_n, \theta_{1:k} \sim \text{Normal}(\theta_{x_n}, \lambda^2).$$

- To keep things simple, we assume that λ^2 is known and is the same for all options. It is also possible to allow λ^2 to be unknown, and to vary with x.
- At time N, we select an option $\hat{x} \in \{1, \ldots, k\}$, which we hope is the best option.
- We receive a reward of $\theta_{\hat{x}}$, which is the true value of the selected option $\hat{x}$.

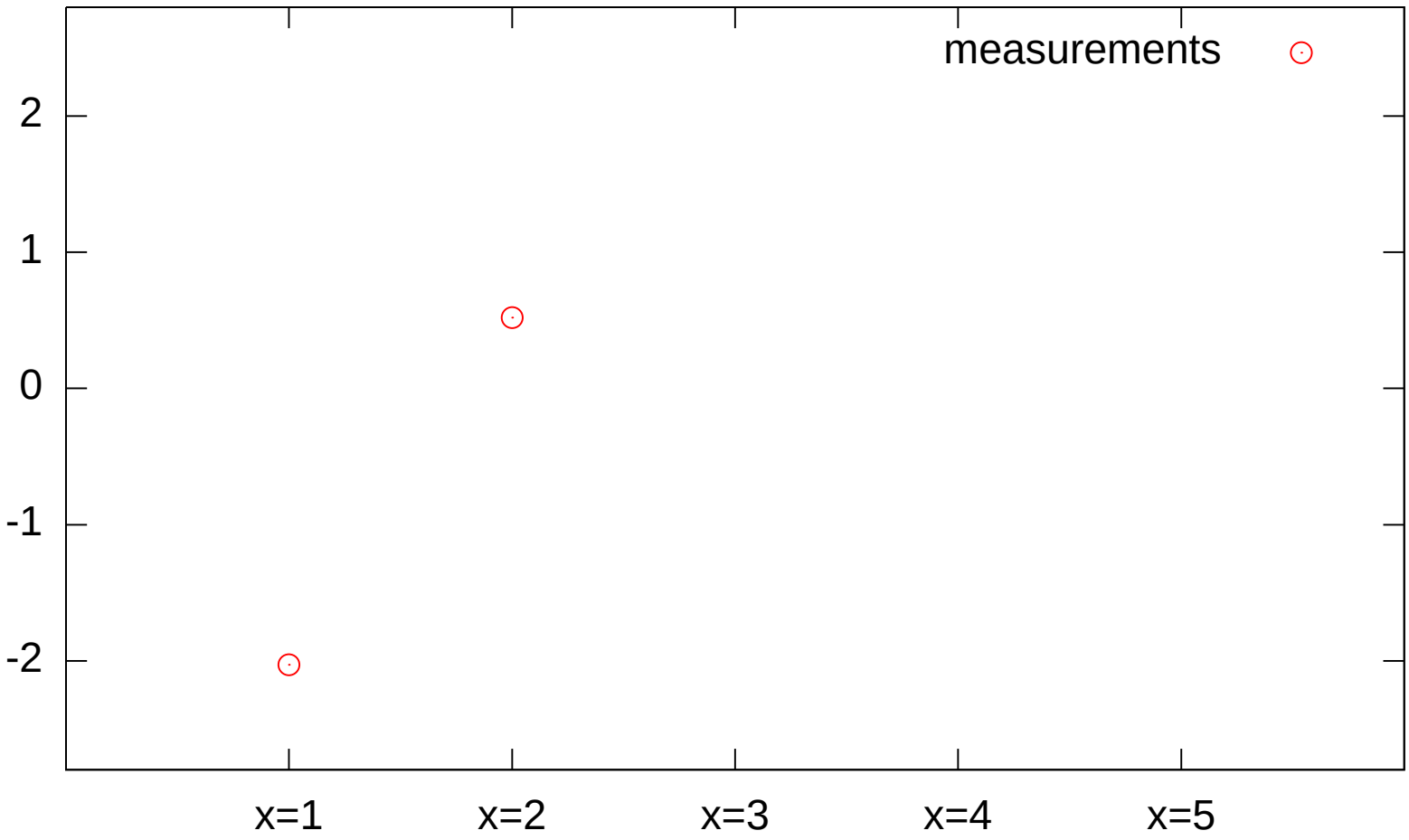
Example, Time 0



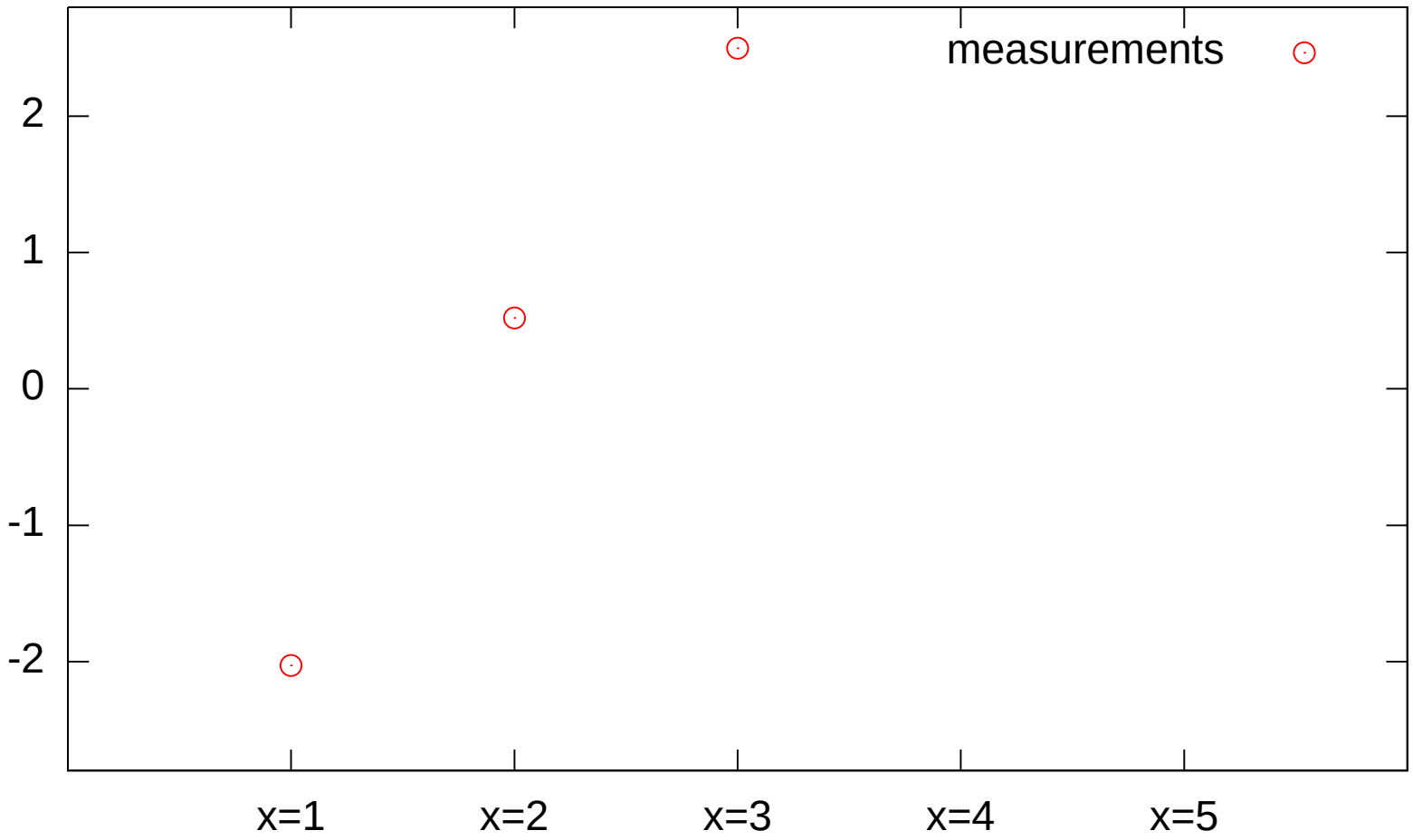
Example, Time 1



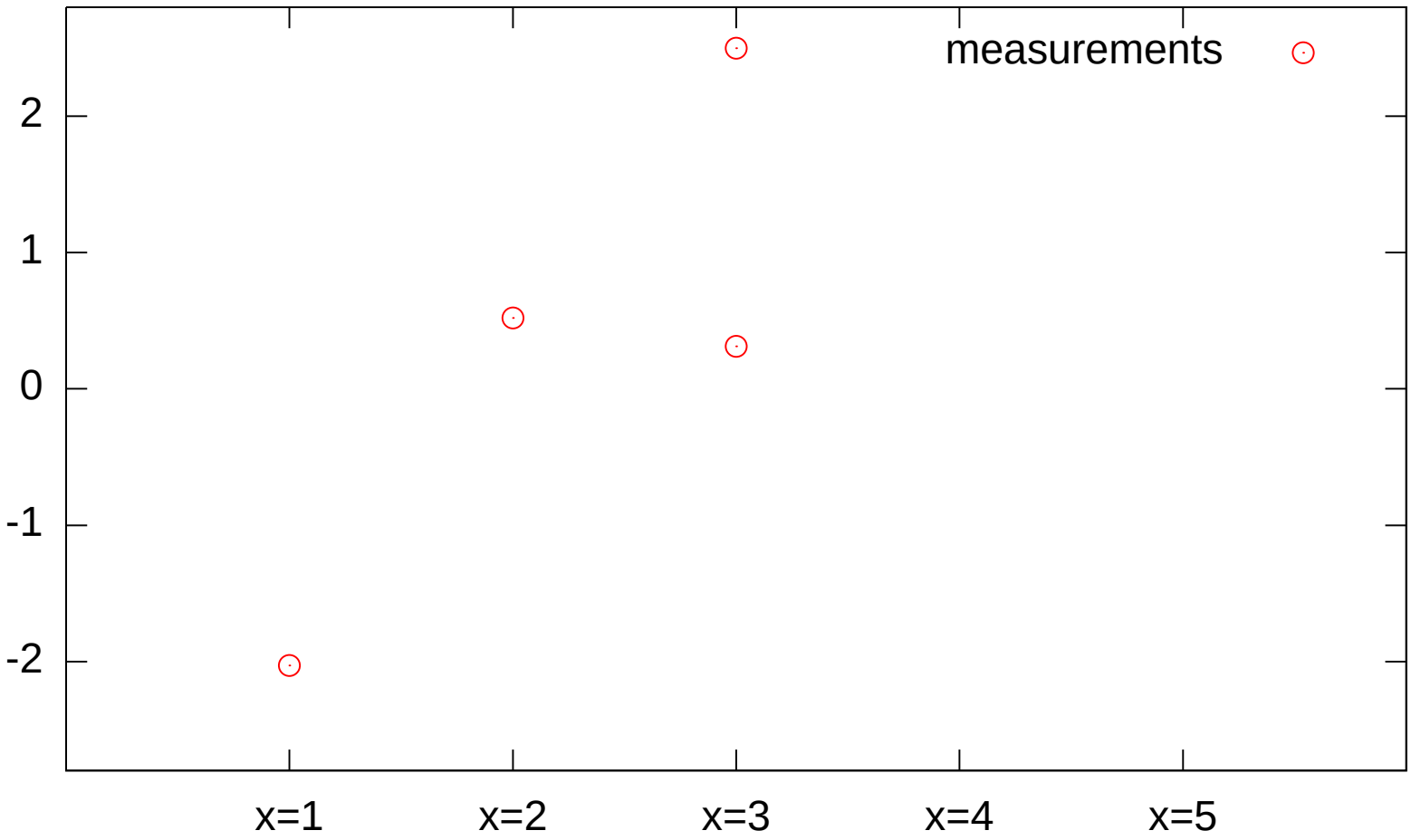
Example, Time 2



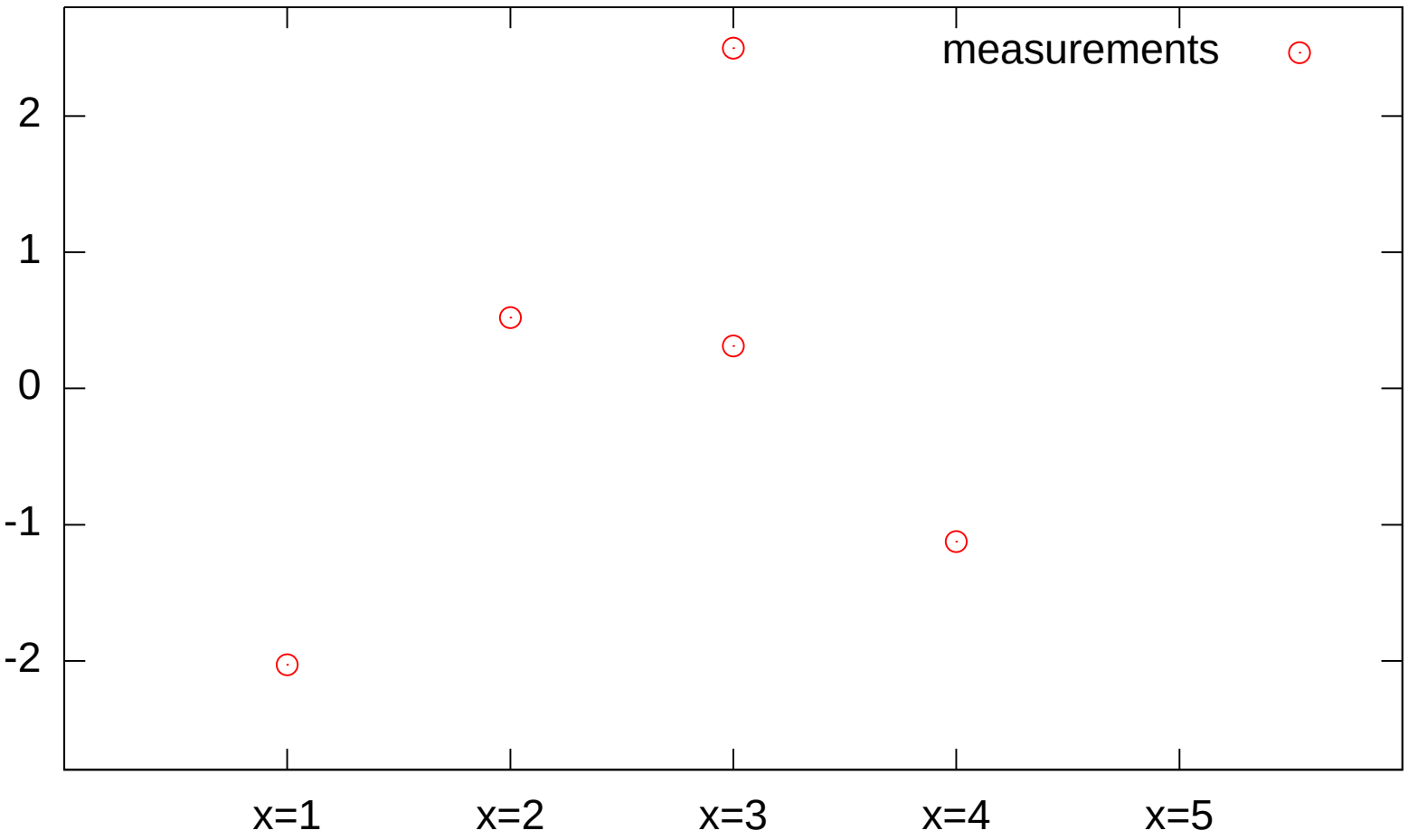
Example, Time 3



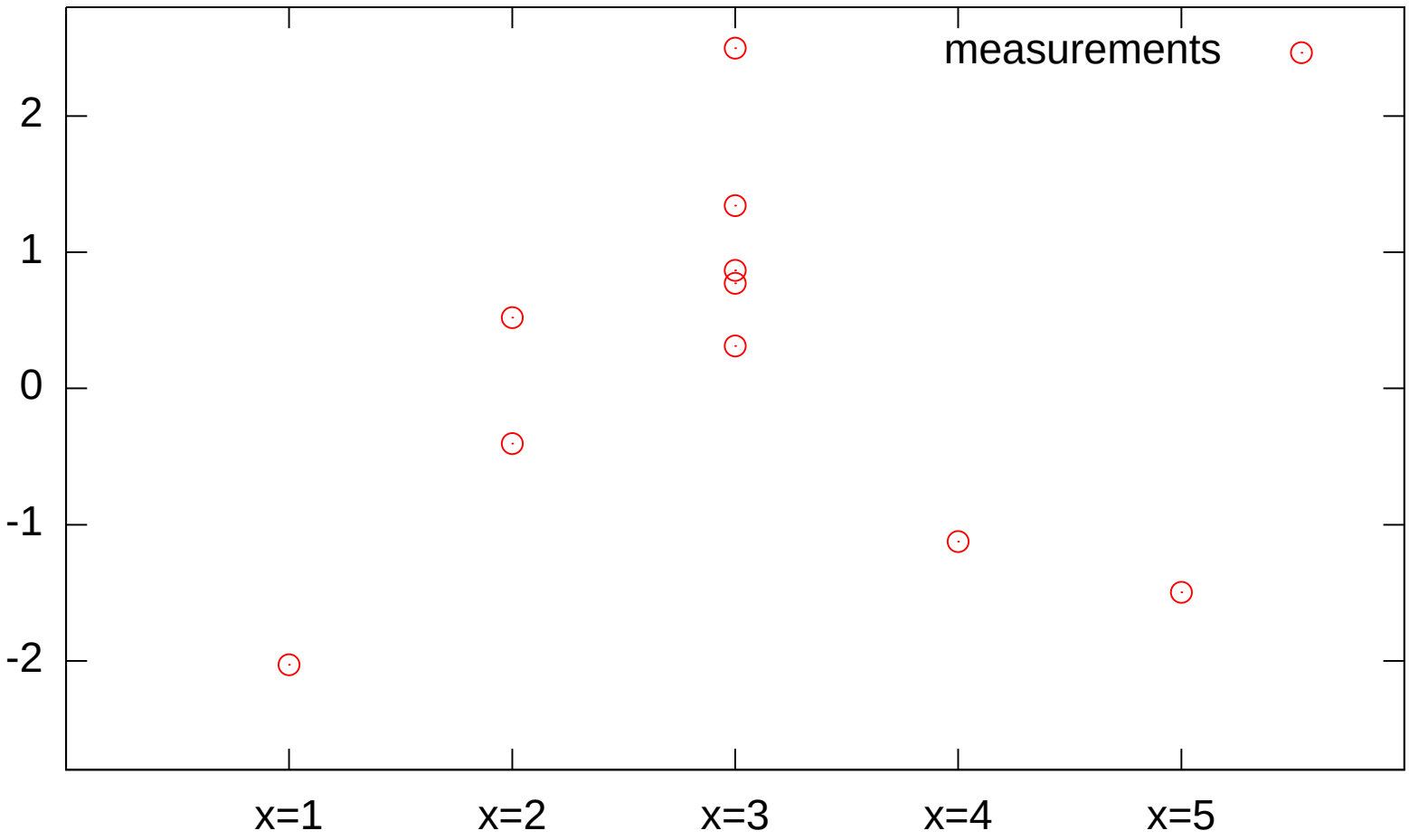
Example, Time 4



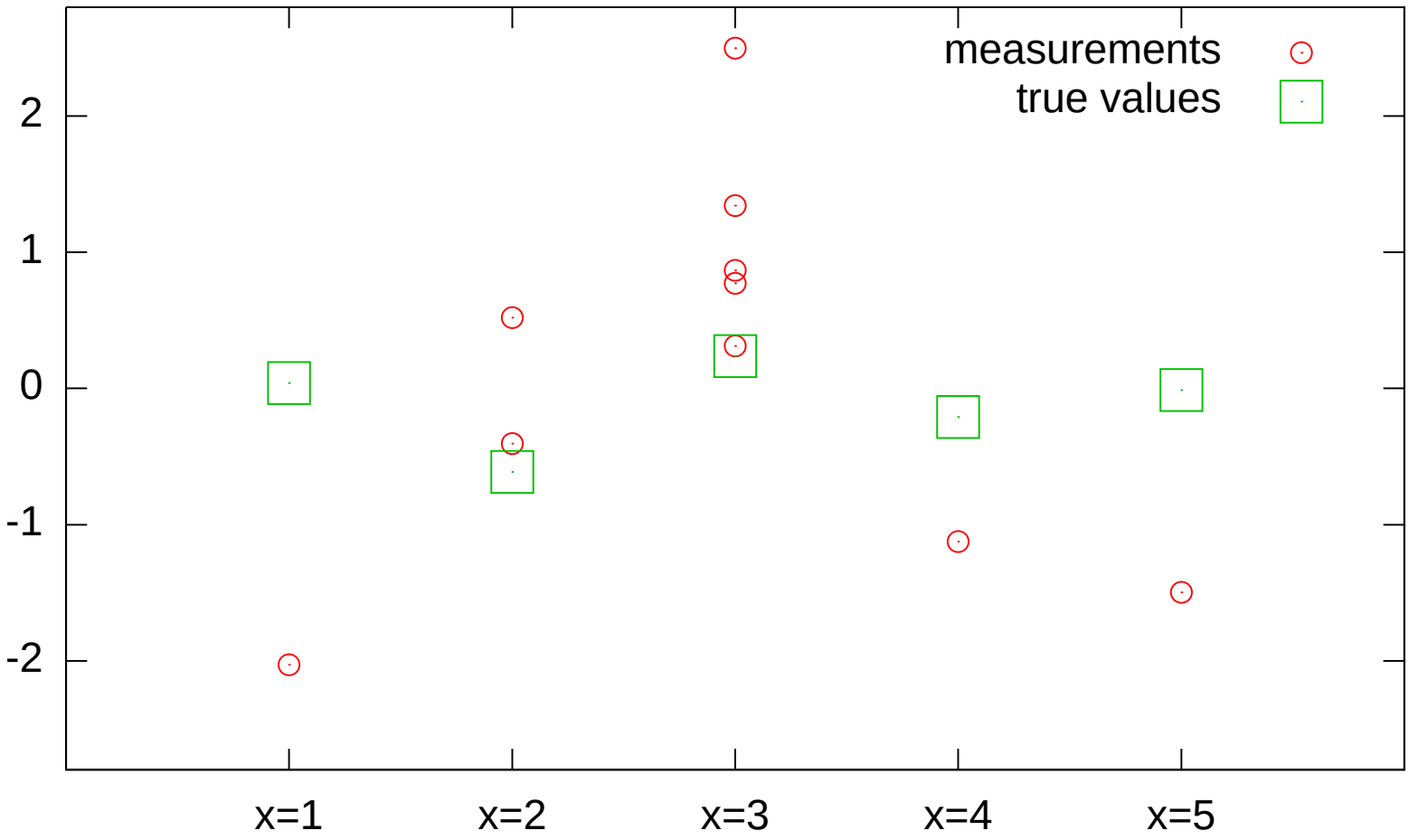
Example, Time 5



Example, Time 10



Example, Time 10



Outline

We put a Bayesian prior probability distribution on θ

- The underlying value of alternative x is θ_x.
- We do not know θ_x, but based on intuition, experience, and data from other similar problems, we may be able to make statements like "The expected waiting time for this nurse staffing policy is probably between 15 minutes and 4 hours."
- We formalize this by supposing that θ_x was drawn by nature at random from a *Bayesian prior probability distribution*.
- Once θ_x is drawn by nature (at time $n = 0$), it stays fixed (over $n = 1, 2, \ldots$).
- We use a normal prior probility distribution, because it is flexible, and allows easy computation:

$$\theta_x \sim \text{Normal}(\mu_{0x}, \sigma^2_{0x}).$$

We can use Bayesian statistics to estimate θ_x, based on noisy samples.

- Suppose our first sample is from option x, so $x_1 = x$.
- We observe
$$y_1 \mid x_1 = x, \theta_{1:k} \sim \text{Normal}(\theta_x, \lambda^2).$$
- We can use *Bayes rule* to calculate the conditional distribution of θ_x given this sample.
- The conditional distribution given the data is called the *posterior distribution*.

We can use Bayesian statistics to estimate θ_x, based on noisy samples.

- Bayes rule shows us that the posterior distribution on θ_x is

$$\theta_x \mid x_1, y_1 \sim \text{Normal}\left(\mu_{1,x}, \sigma^2_{1,x}\right),$$

where

$$\mu_{1,x} = \frac{(\sigma_{0,x})^{-2}\mu_{0,x} + \lambda^{-2} y_1}{\sigma_{0,x}^{-2} + \lambda^{-2}}$$

$$\sigma^2_{1,x} = \left[\sigma_{0,x}^{-2} + \lambda^{-2}\right]^{-1}$$

- The posterior distribution on $\theta_{x'}$, where $x' \neq x$, does not change.

There is a nice expression for the posterior distribution

In general,

$$\theta_x \mid x_1, \ldots, x_n, y_1, \ldots, y_n \sim \text{Normal}\left(\mu_{n,x}, \sigma^2_{n,x}\right),$$

where $\mu_{n,x}, \sigma_{n,x}$ can be computed recursively.
For $x = x_n$, the posterior is updated via:

$$\mu_{n+1,x} = \frac{\sigma^{-2}_{n,x}\mu_{n,x} + \lambda^{-2} y_{n+1}}{\sigma^{-2}_{n,x} + \lambda^{-2}}$$

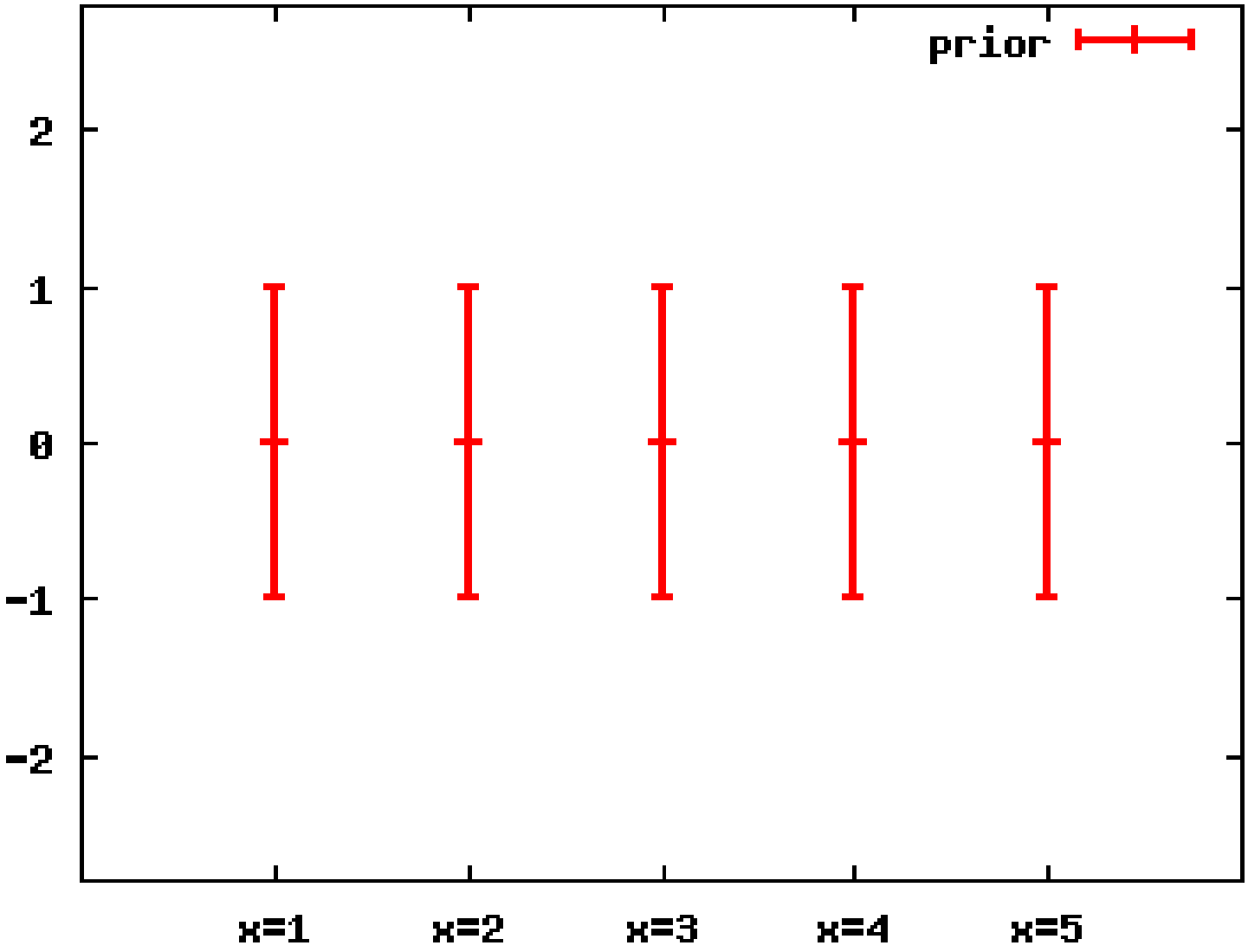
$$\sigma^2_{n+1,x} = \left[\sigma^{-2}_{n,x} + \lambda^{-2}\right]^{-1}$$

and the posterior for $x \neq x_n$ does not change:

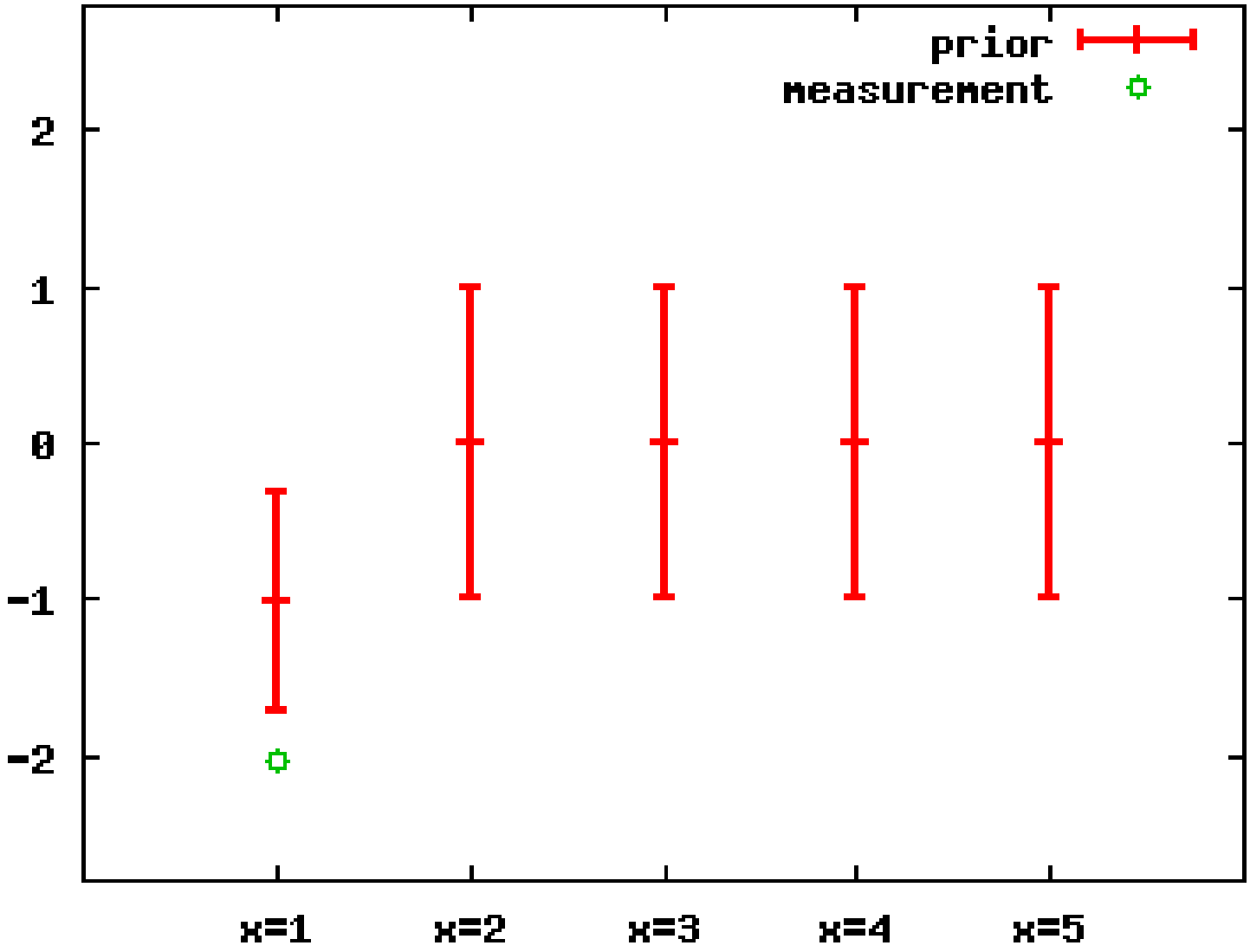
$$\mu_{n+1,x} = \mu_{n,x} \text{ for } x \neq x^n$$

$$\sigma_{n+1,x} = \sigma_{n,x} \text{ for } x \neq x^n$$

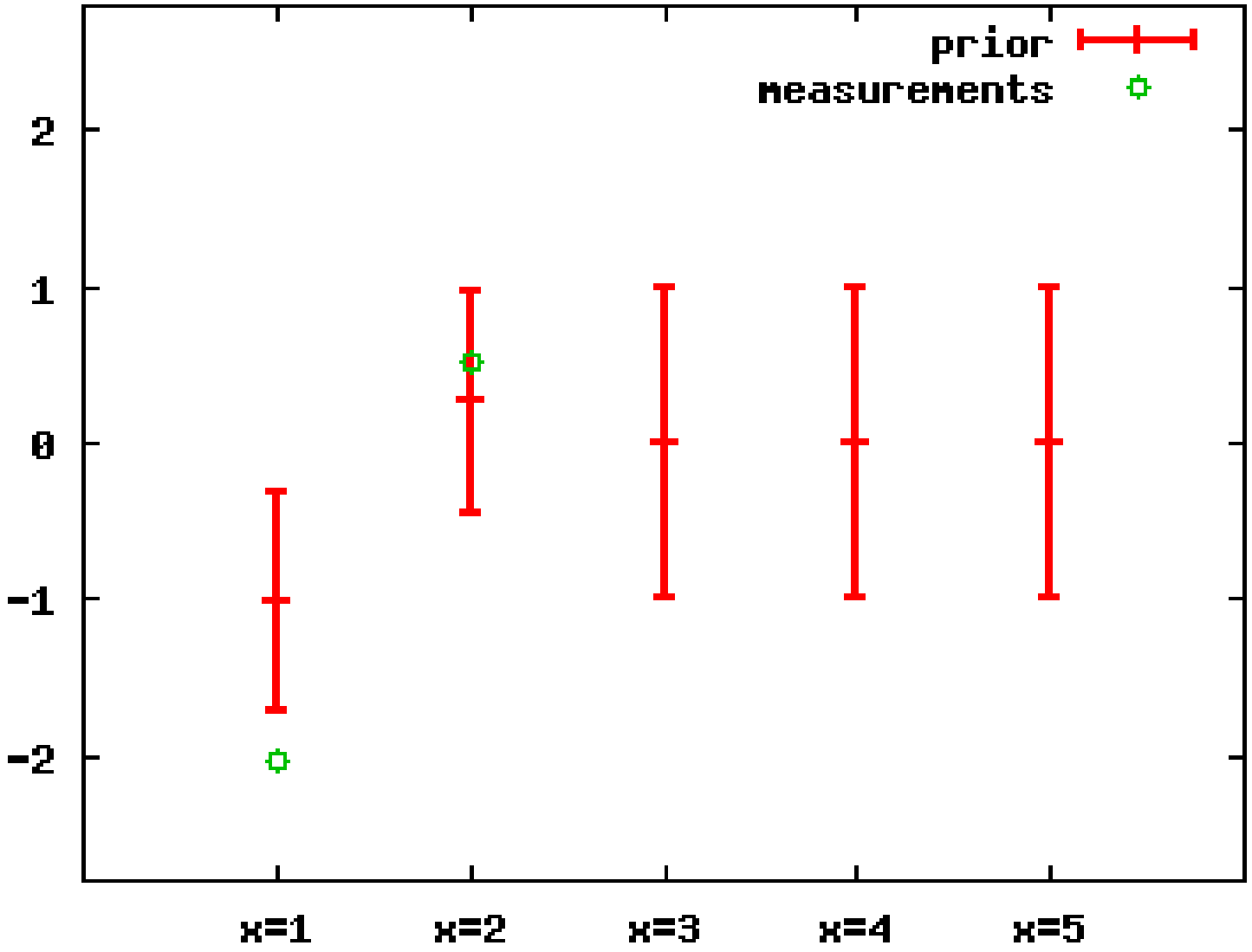
Example of the posterior distribution



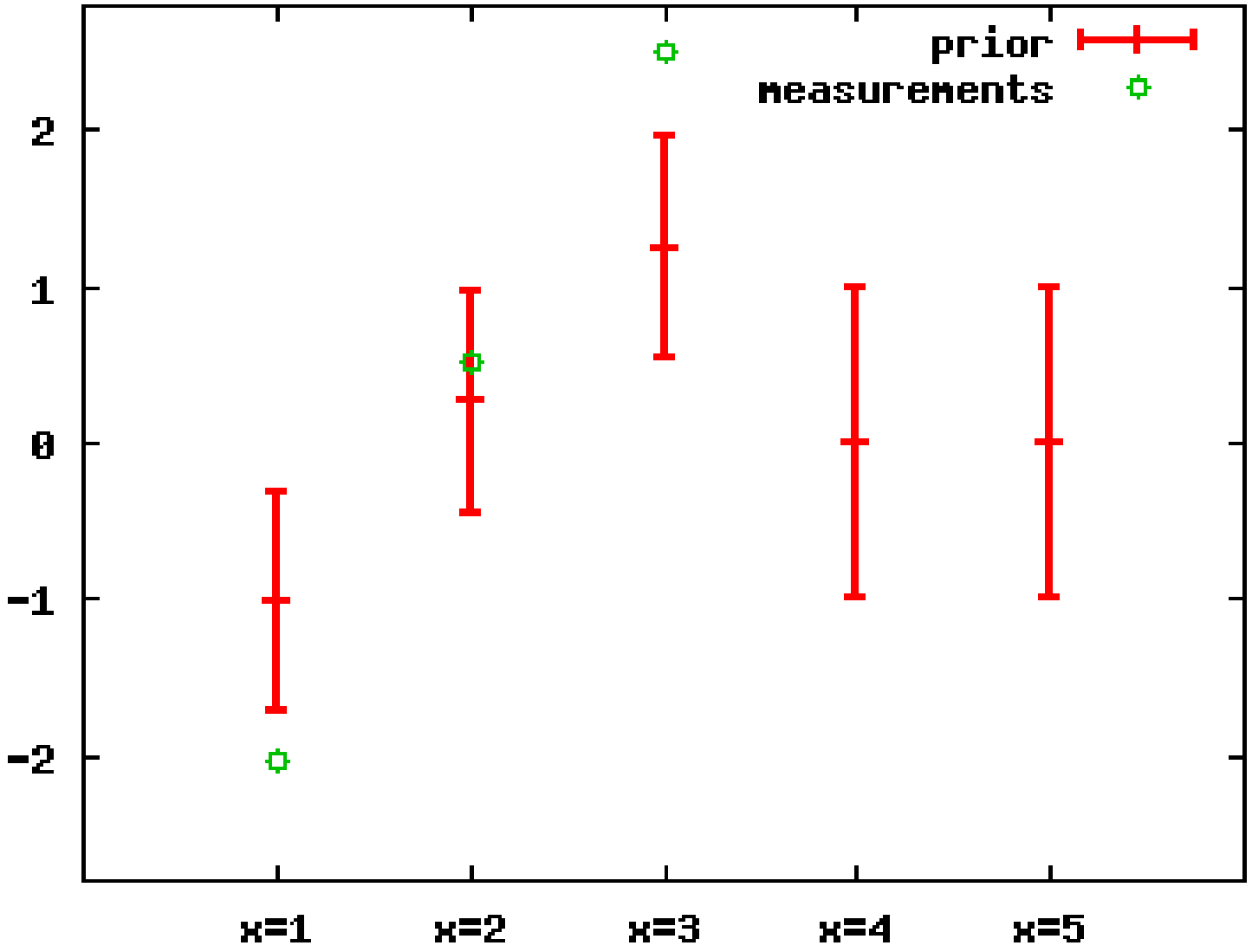
Example of the posterior distribution



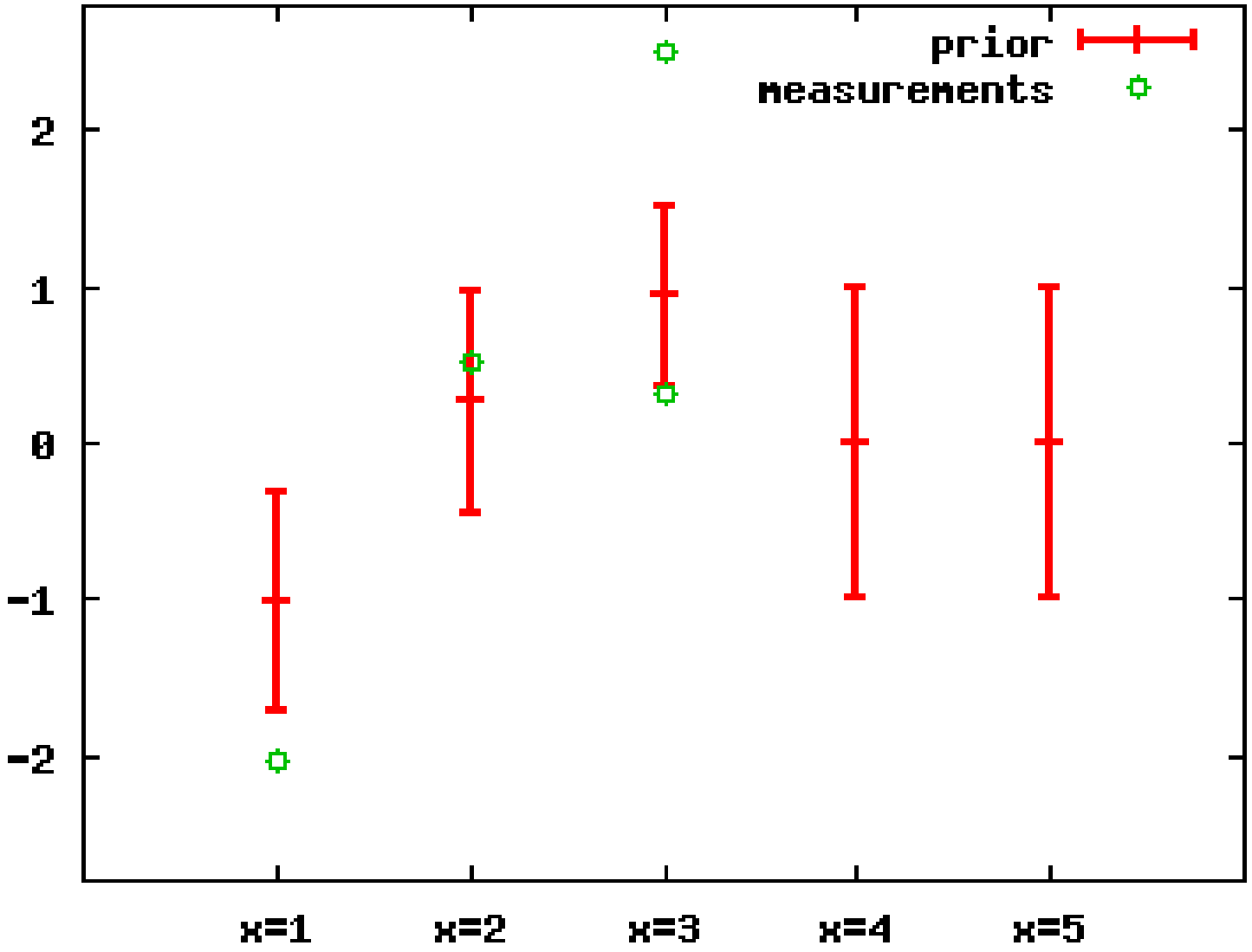
Example of the posterior distribution



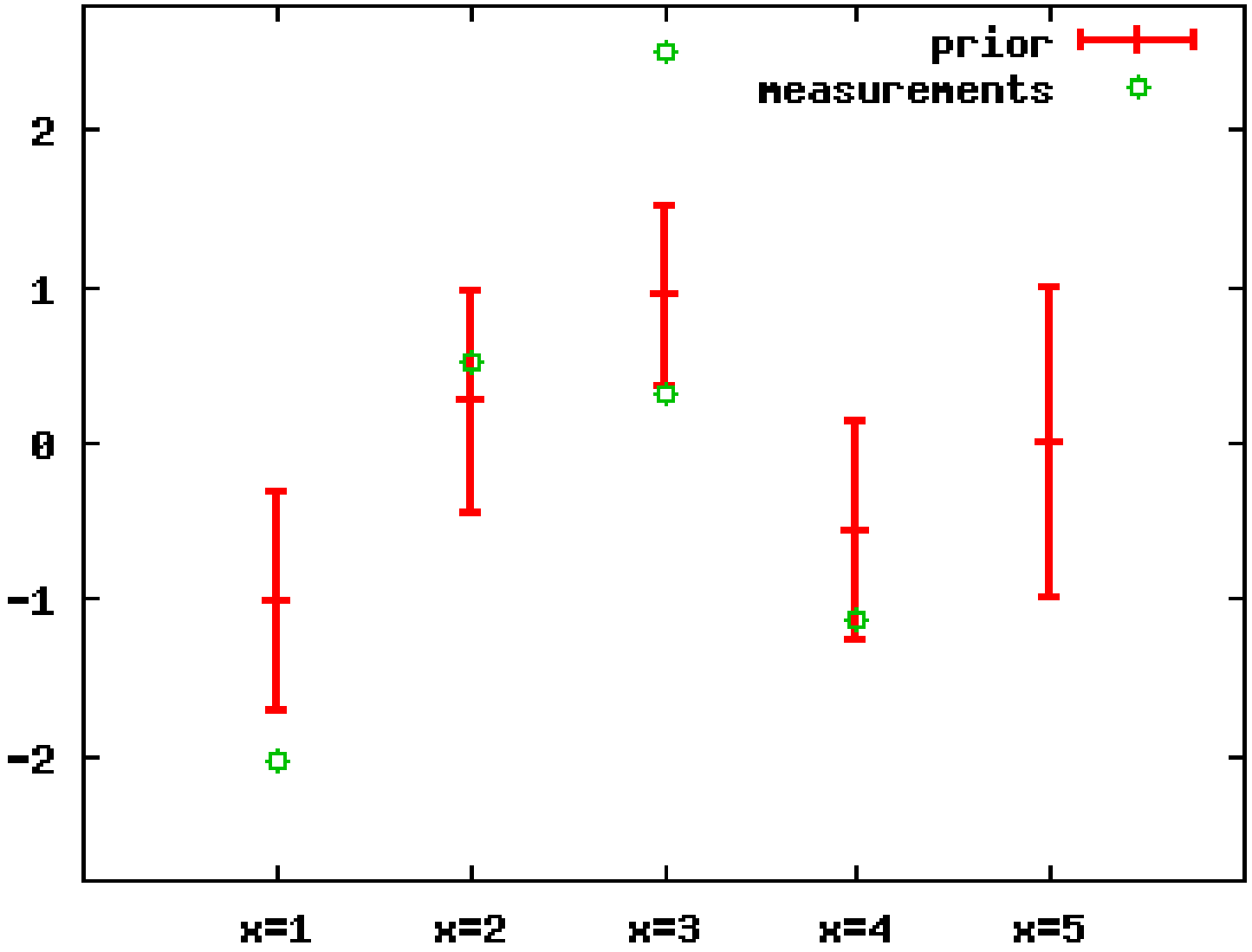
Example of the posterior distribution



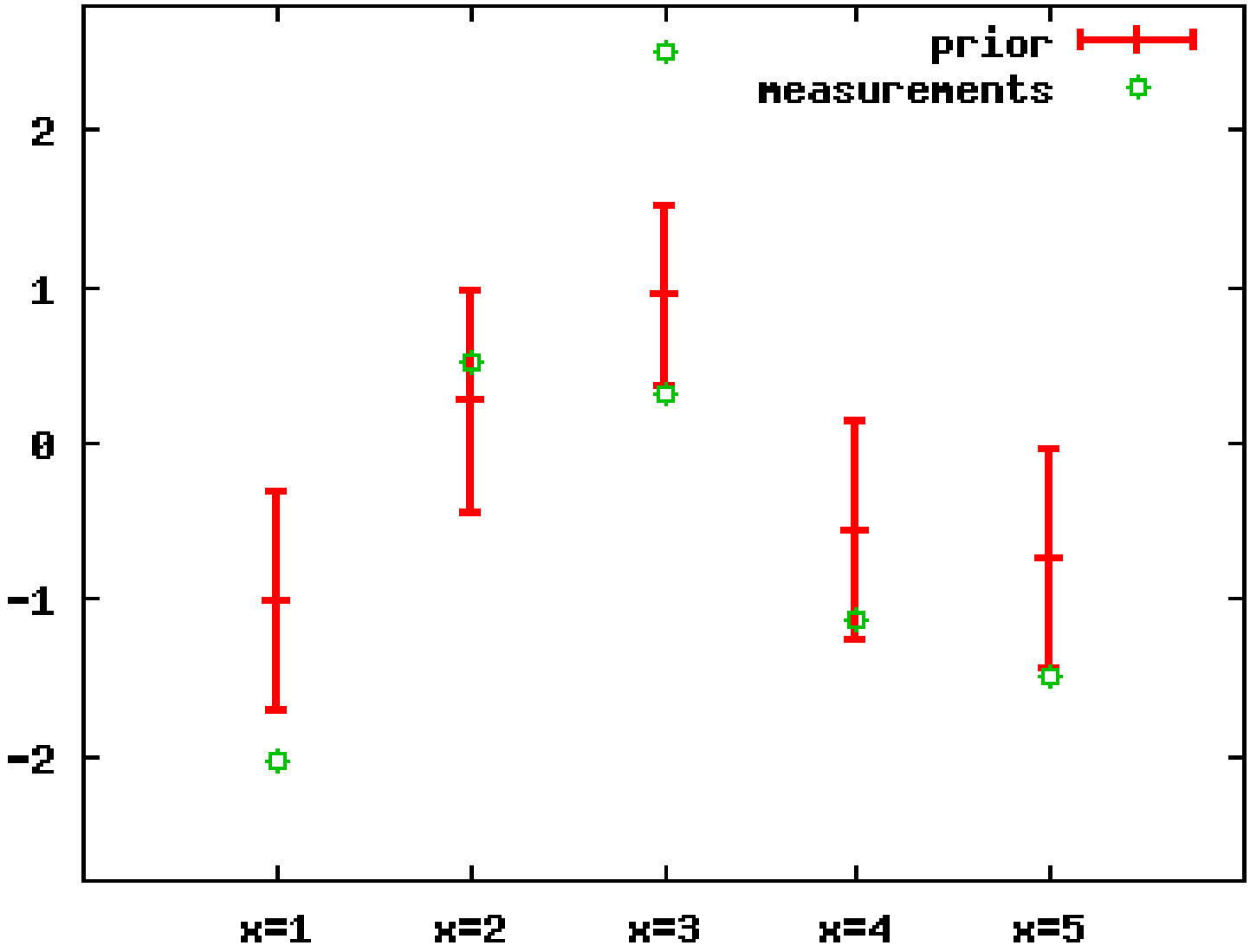
Example of the posterior distribution



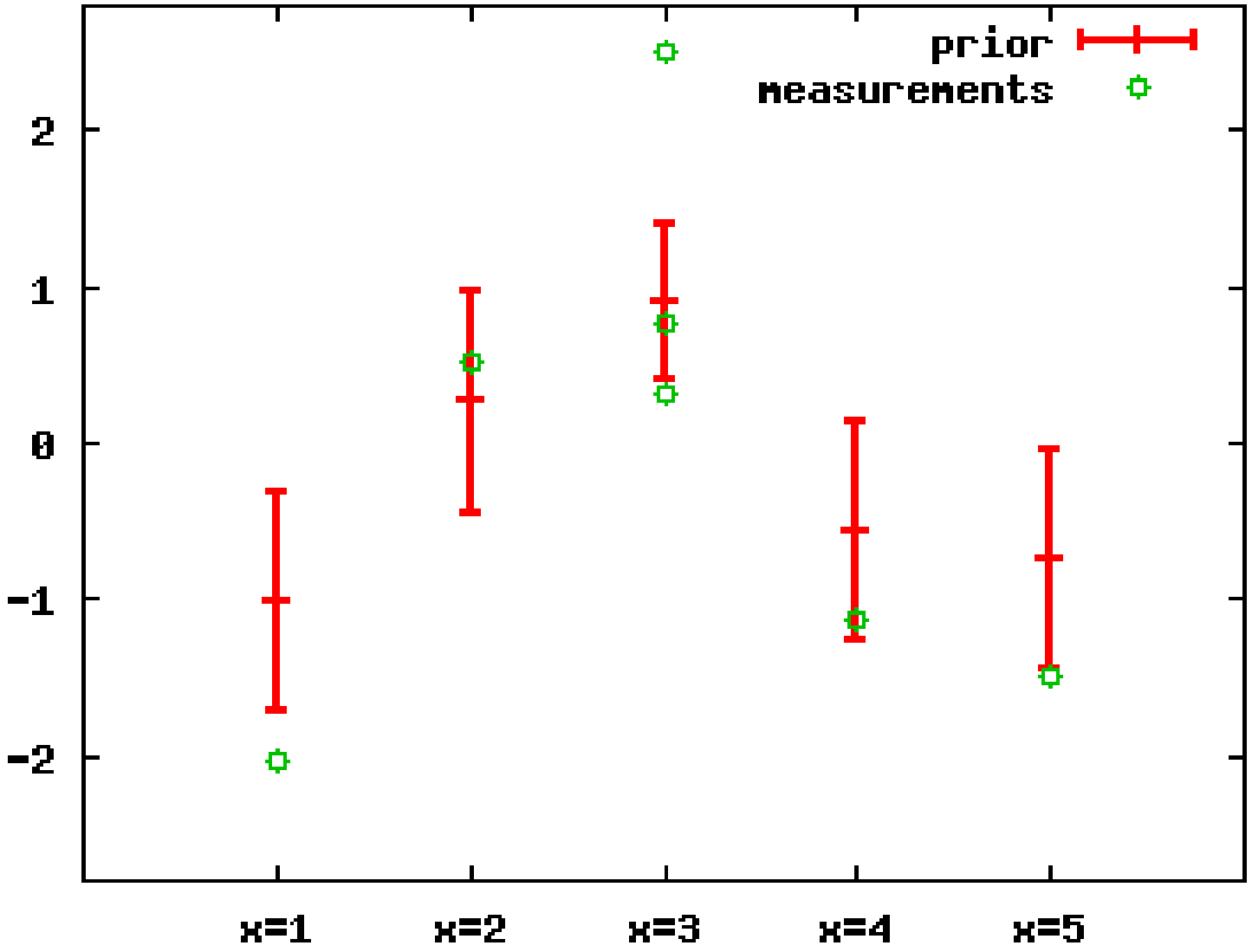
Example of the posterior distribution



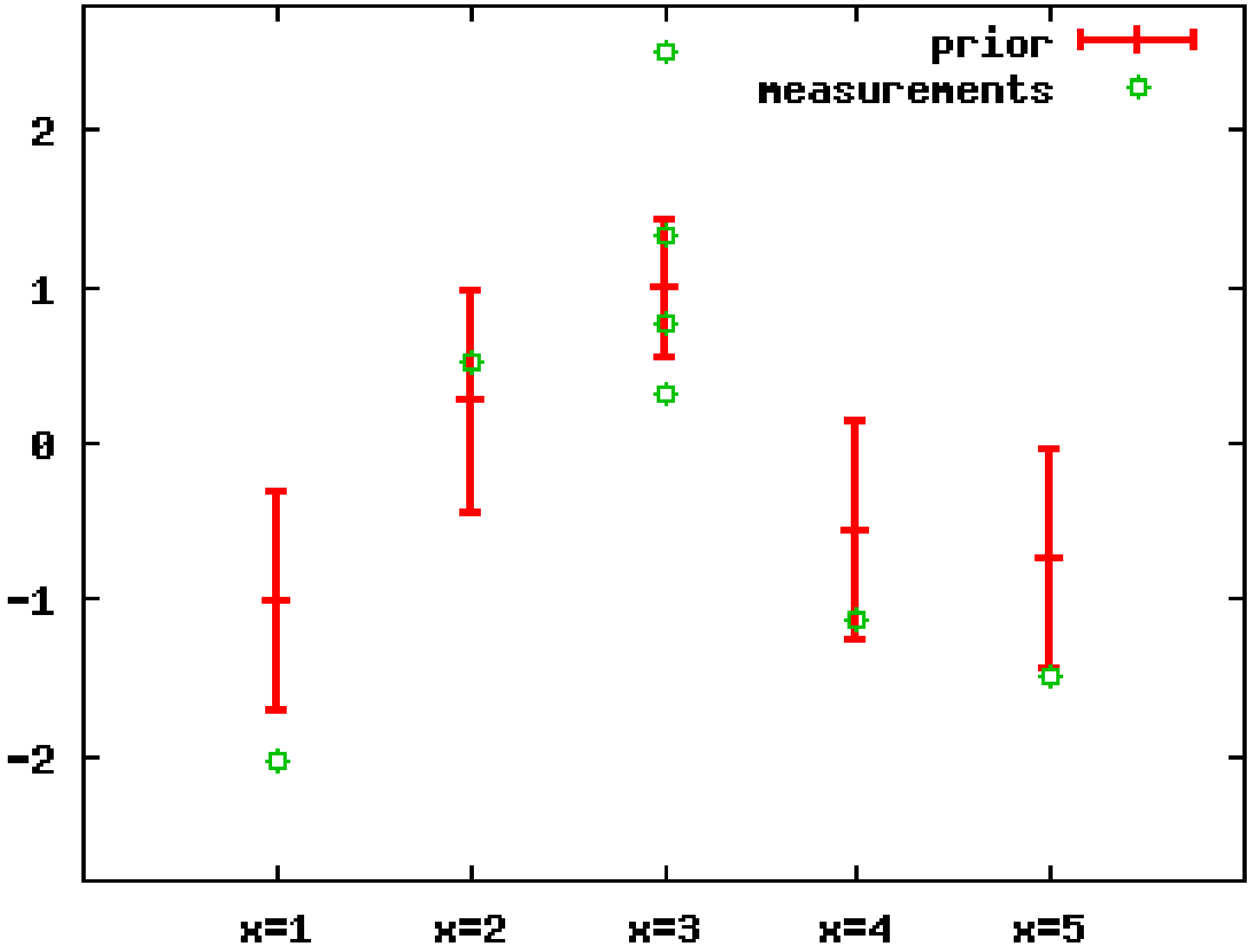
Example of the posterior distribution



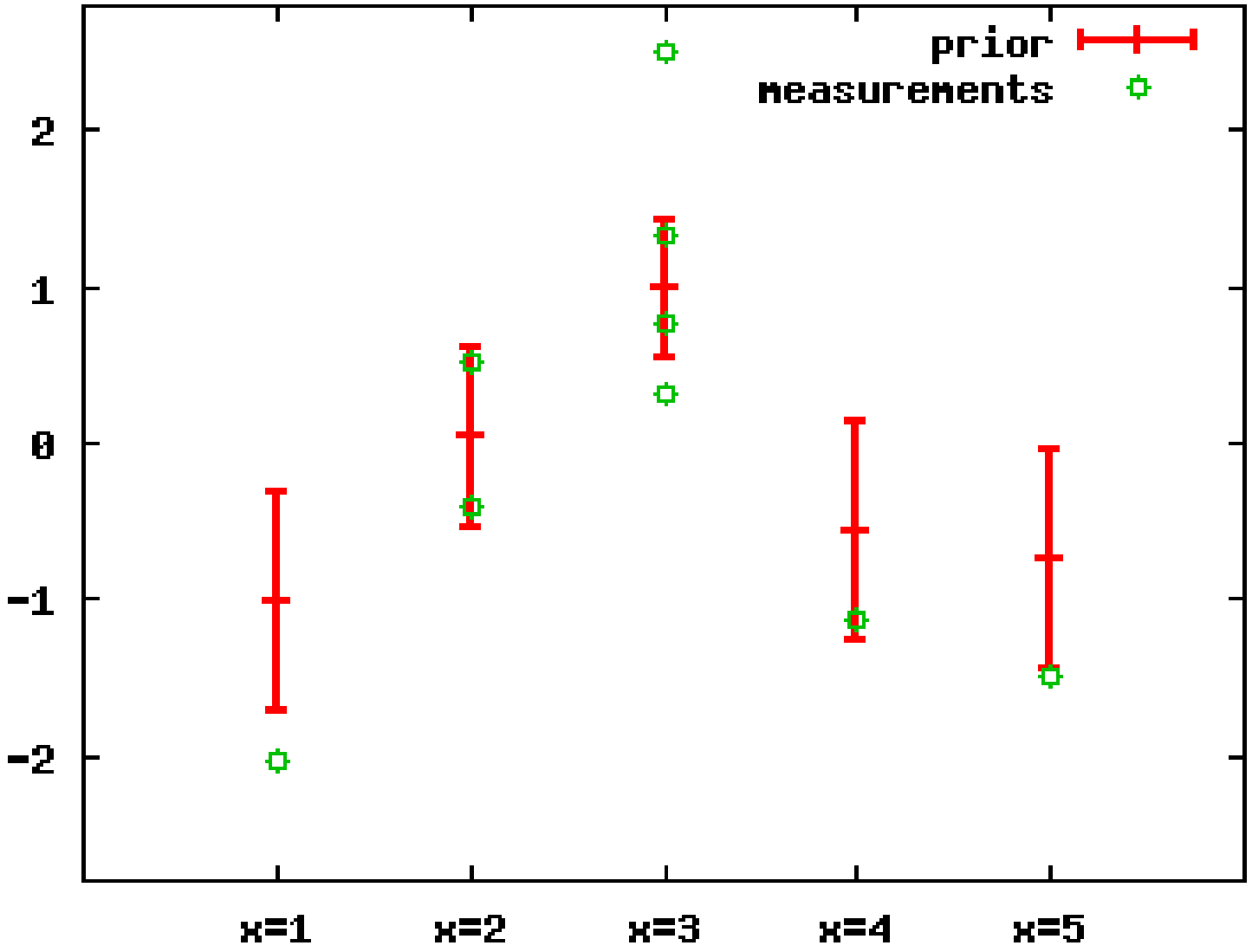
Example of the posterior distribution



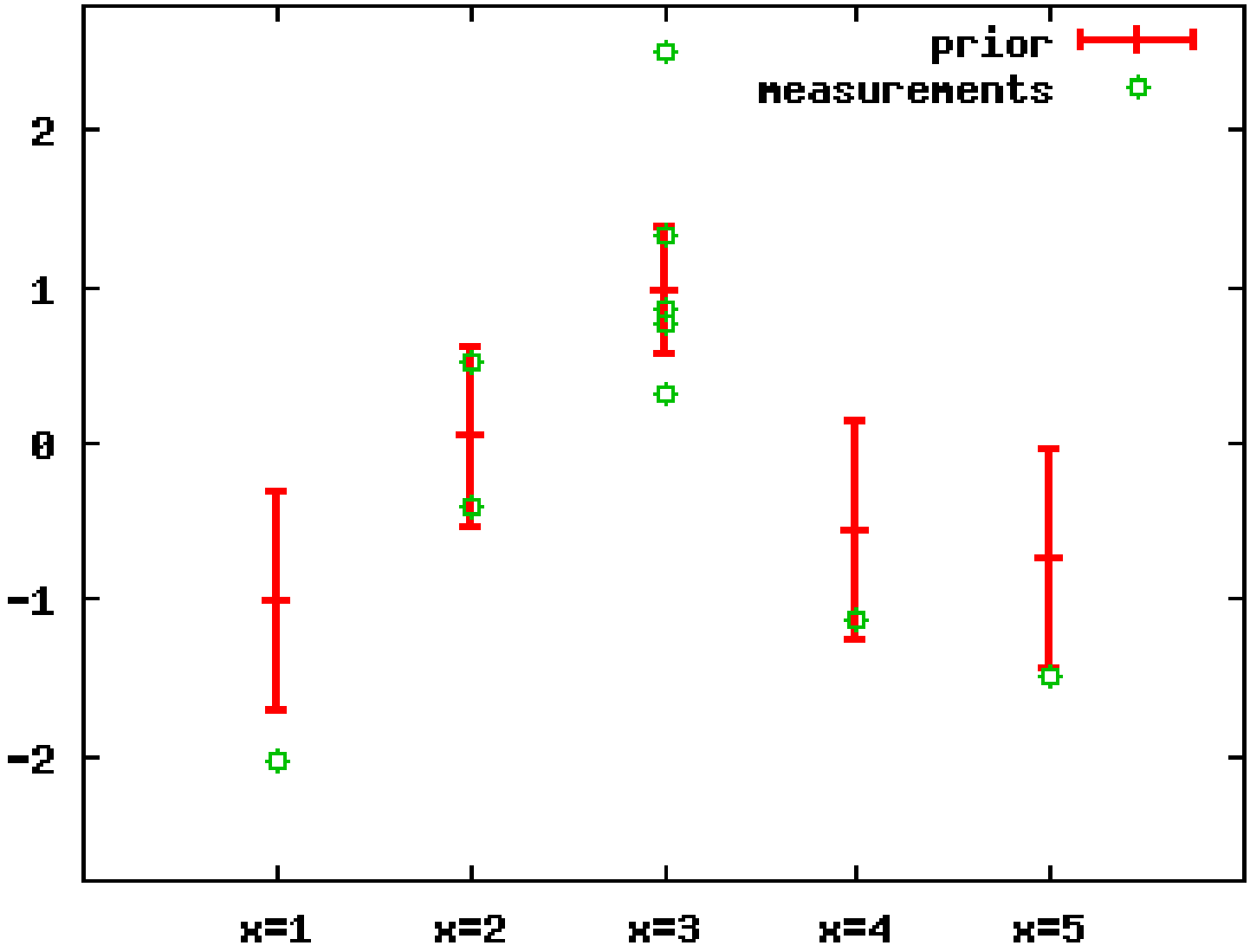
Example of the posterior distribution



Example of the posterior distribution



Example of the posterior distribution



We can use the posterior distribution to choose $\hat{x}$

- Recall that $\hat{x}$ is our selection of the best, and it is chosen at time N based on all previous samples $x_1, \ldots, x_N, y_1, \ldots, y_N$.
- Based on these samples, the posterior is,

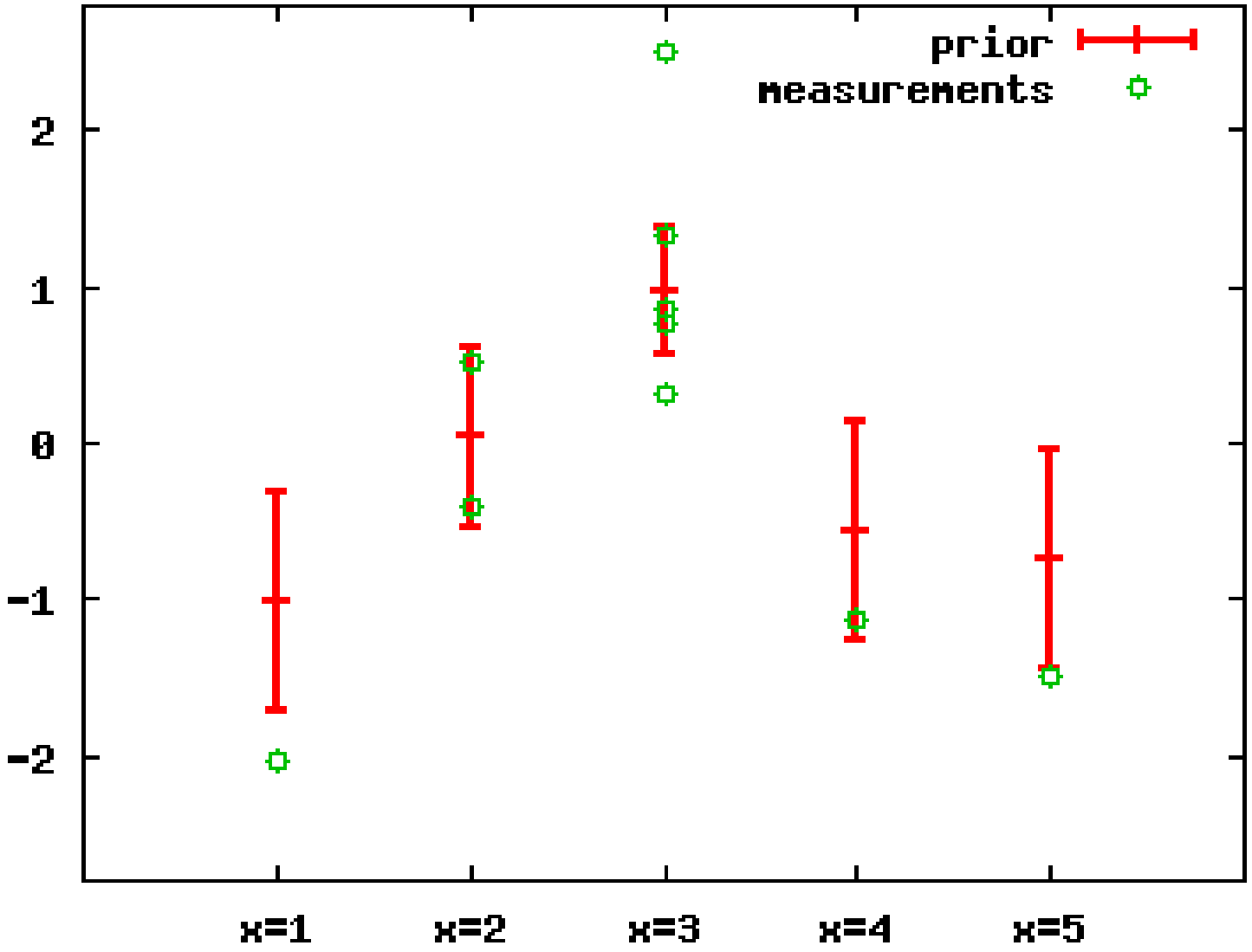
$$\theta_x \mid x_1, \ldots, x_N, y_1, \ldots, y_N \sim \text{Normal}\left(\mu_{N,x}, \sigma^2_{N,x}\right).$$

- Recall that the reward for choosing $\hat{x} = x$ is θ_x.
- The conditional expected reward for choosing $\hat{x} = x$ is

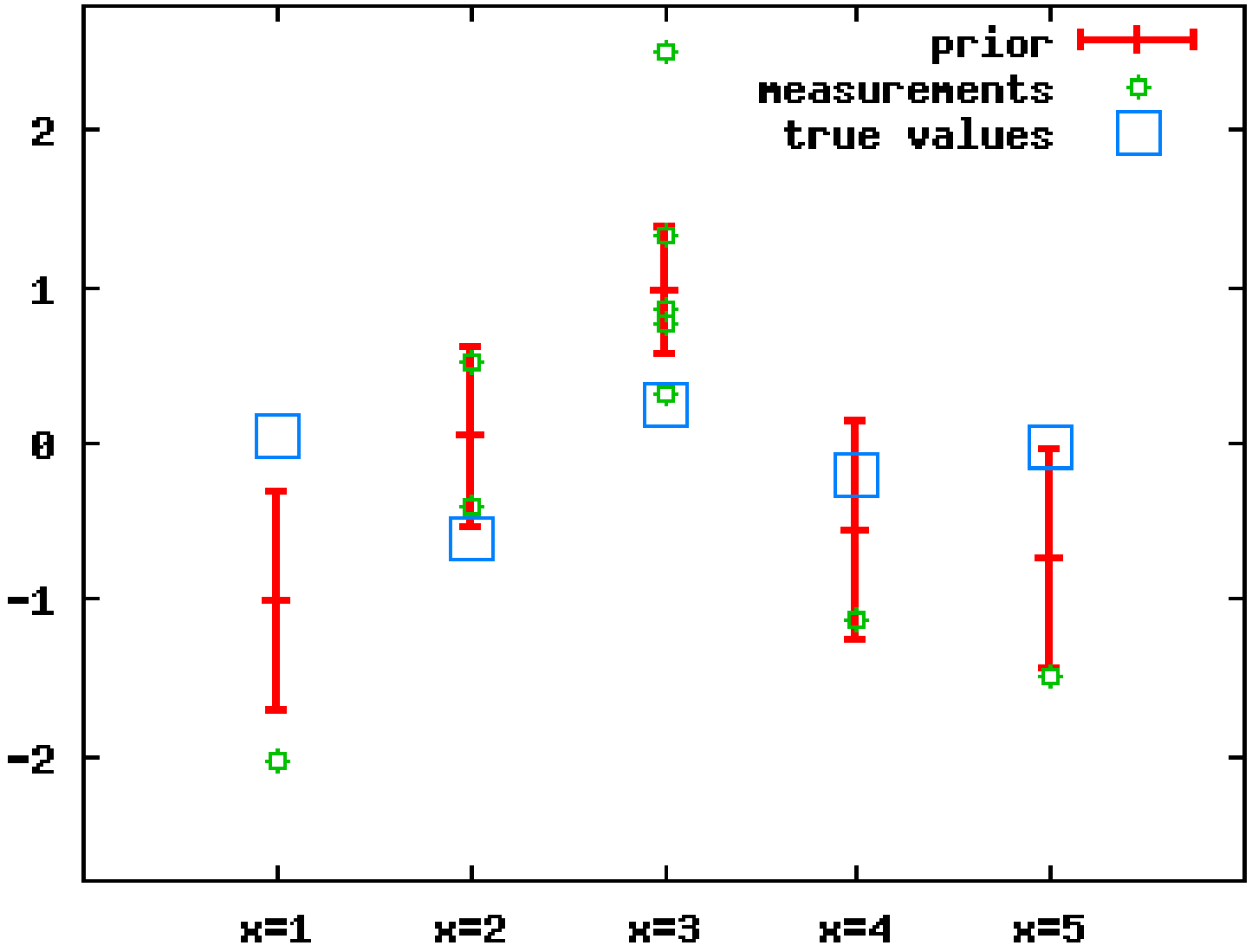
$$E\left[\theta_x \mid x_1, \ldots, x_N, y_1, \ldots, y_N\right] = \mu_{N,x}.$$

- Thus, the choice that gives the biggest conditional expected reward is $\arg\max_x \mu_{N,x}$ and it has value $\max_x \mu_{N,x}$.

Example of choosing $\hat{x}$



Example of choosing $\hat{x}$



How should we choose the x_n?

- Our ability to choose $\hat{x}$ accurately depends on the chooses we make for $x_1, \ldots, x_N$.
- Intuitively, a good way to choose these should spend the first part of the budget exploring the options to figure out which ones are among the best, and then focus the rest of the budget on these options.
- But how precisely should we accomplish this?
- One way to choose the x_n is through the **knowledge-gradient (KG) method** for independent beliefs.
- Later, in the seminar, I will talk about the knowedge-gradient method for correlated beliefs.

Outline

The knowledge-gradient factor quantifies a sample's value

- The knowledge-gradient method is created via the following thought experiment.
- If we were to stop at time n, and select $\hat{x}$ based on $x_{1:n}, y_{1:n}$, we would earn an expected reward of

$$\mu_n^* = \max_x \mu_{n,x}.$$

- If we were to take one more sample, x_{n+1}, and observe y_{n+1}, and then select $\hat{x}$, we would earn an expected reward of

$$\mu_{n+1}^* = \max_x \mu_{n+1,x}.$$

The knowledge-gradient factor quantifies a sample's value

- Before the new sample, our value was μ_n^*. After, it was μ_{n+1}^*.
- The additional sample x_{n+1}, y_{n+1} has increased our solution's value by

$$\mu_{n+1}^* - \mu_n^*.$$

- At time n, we don't know y_{n+1}, so we can't compute this quantity.
- We can, however, compute its expected value,

$$\mathrm{KG}_n(x) = \mathbb{E}_n \left[\mu_{n+1}^* - \mu_n^* \mid x_{n+1} = x \right].$$

- We call this quantity the *knowledge-gradient (KG) factor*, because it measures the change in the value of our knowledge.

Computing the KG factor requires us to think about how the next measurement will change our posterior.

- At time n, suppose we decide to measure $x_{n+1} = x$.
- Before we observe y_{n+1}, it is random.
- We can calculate its conditional distribution given $x_1, \ldots, x_{n+1}, y_1, \ldots, y_n$.

$$y_{n+1} \mid x_1, \ldots, x_{n+1}, y_1, \ldots, y_n \sim \text{Normal}(\mu_{n,x}, \sigma^2_{n,x} + \lambda^2),$$

- From this, and the formula for $\mu_{n+1,x}$ in terms of $\mu_{n,x}$, $\sigma^2_{n,x}$, and y_{n+1}, we can calculate that

$$\mu_{n+1,x} \mid x_1, \ldots, x_{n+1}, y_1, \ldots, y_n \sim \text{Normal}(\mu_{n,x}, \tilde{\sigma}^2_{n,x}),$$

  where $\tilde{\sigma}_{n,x} = \sigma^2_{n,x} / \sqrt{\sigma^2_{n,x} + \lambda^2}$.
- This distribution is called the "posterior predictive distribution".

The KG factor has a convenient formula.

The VOI / KG factor for measuring alternative x at time n is

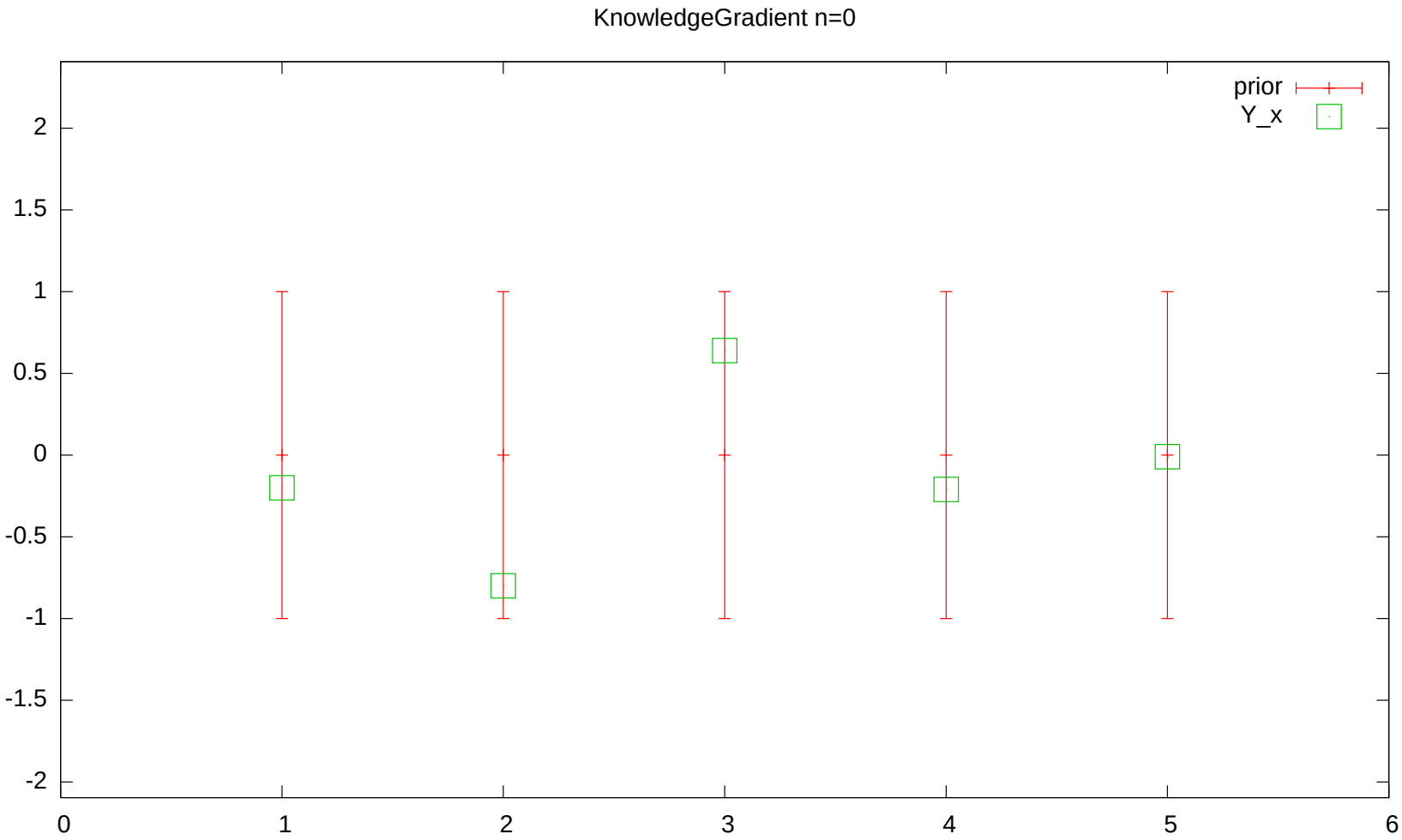
$$\mathrm{KG}_n(x) = \tilde{\sigma}_{n,x} f\left(-\frac{\Delta_{n,x}}{\tilde{\sigma}_{n,x}}\right)$$

where

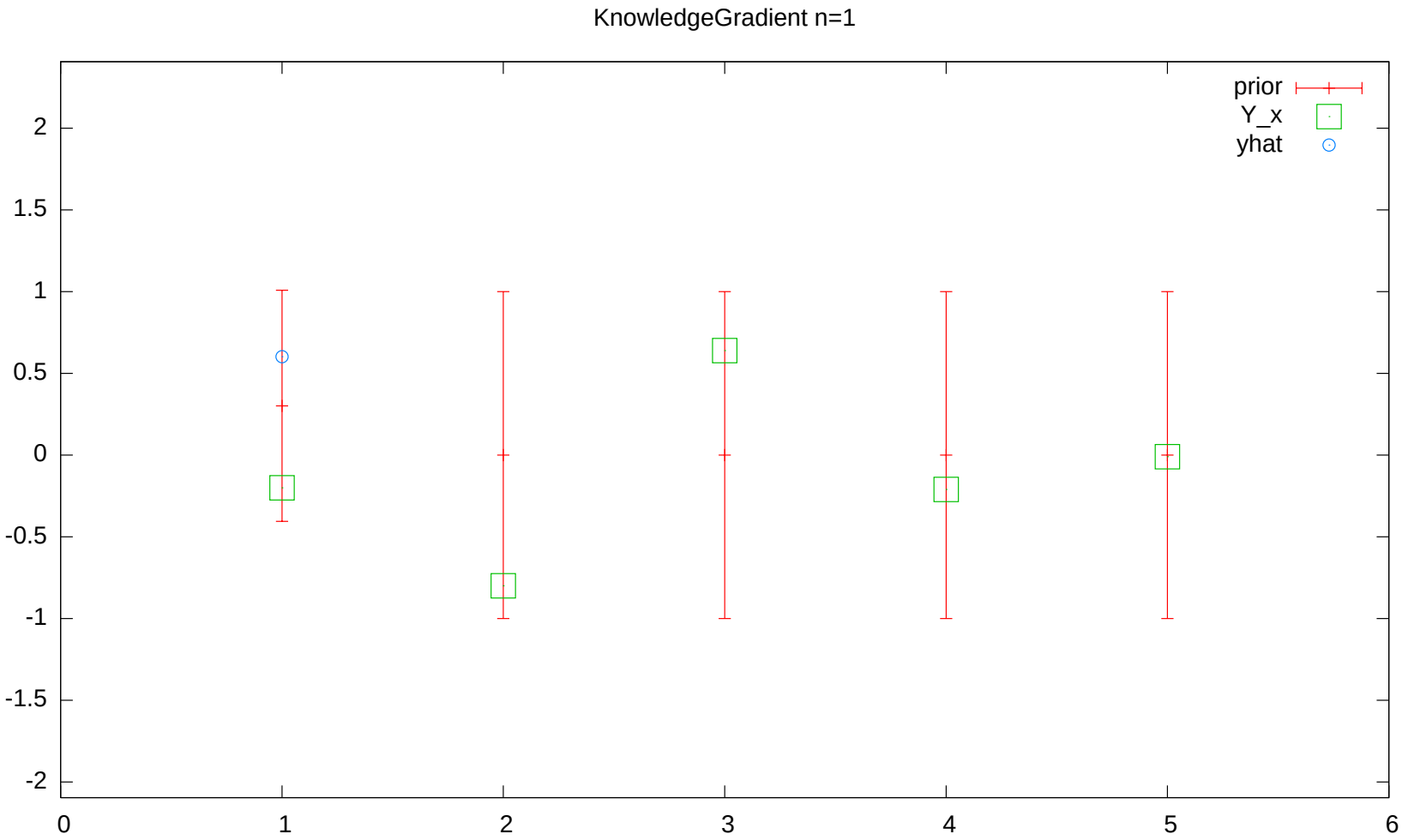
$$\Delta_{n,x} = |\mu_{n,x} - \max_{x' \neq x} \mu_{n,x'}|,$$
$$f(c) = c\Phi(c) + \varphi(c),$$

Φ is the normal cdf, and φ is the normal pdf.

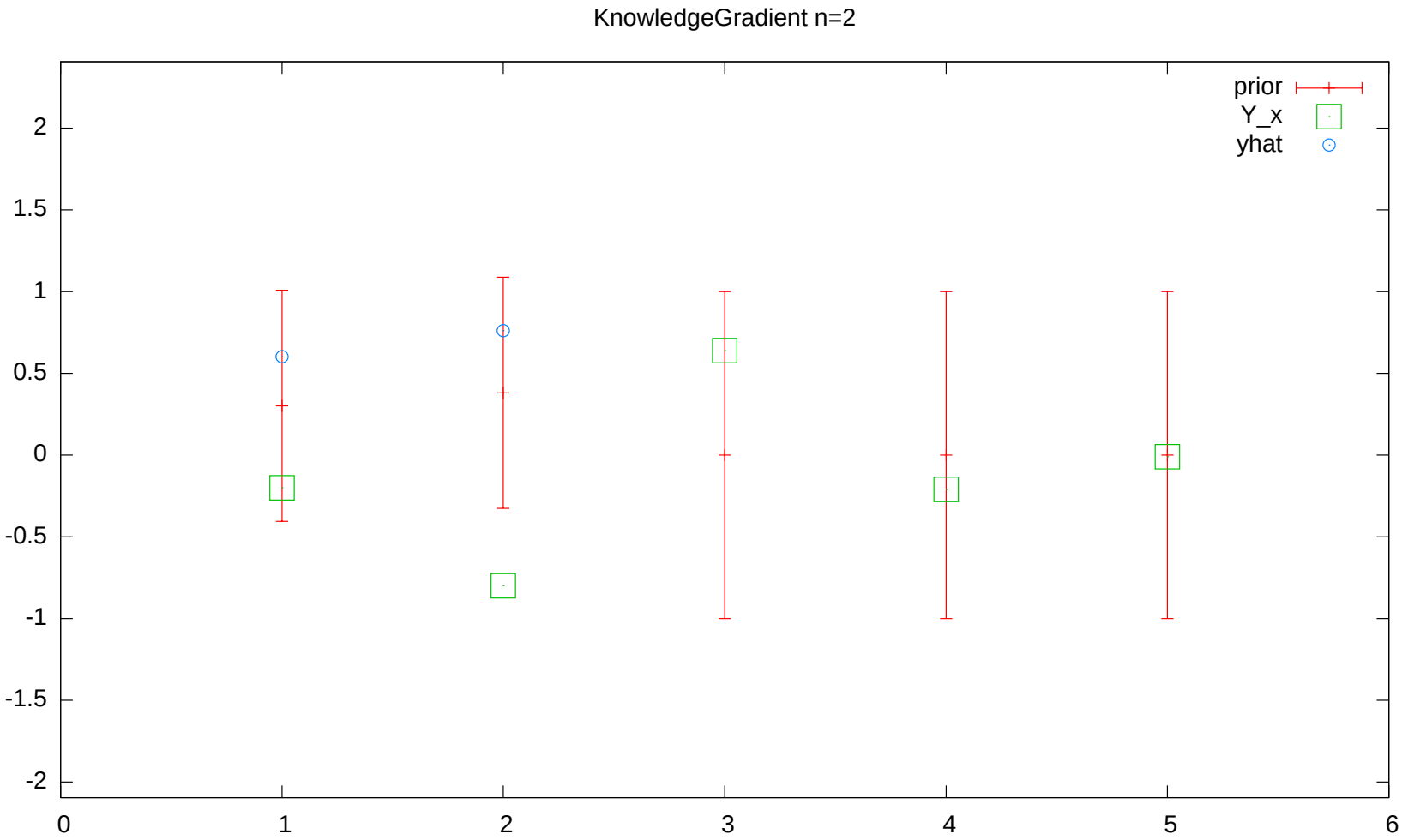
Animation of the KG method



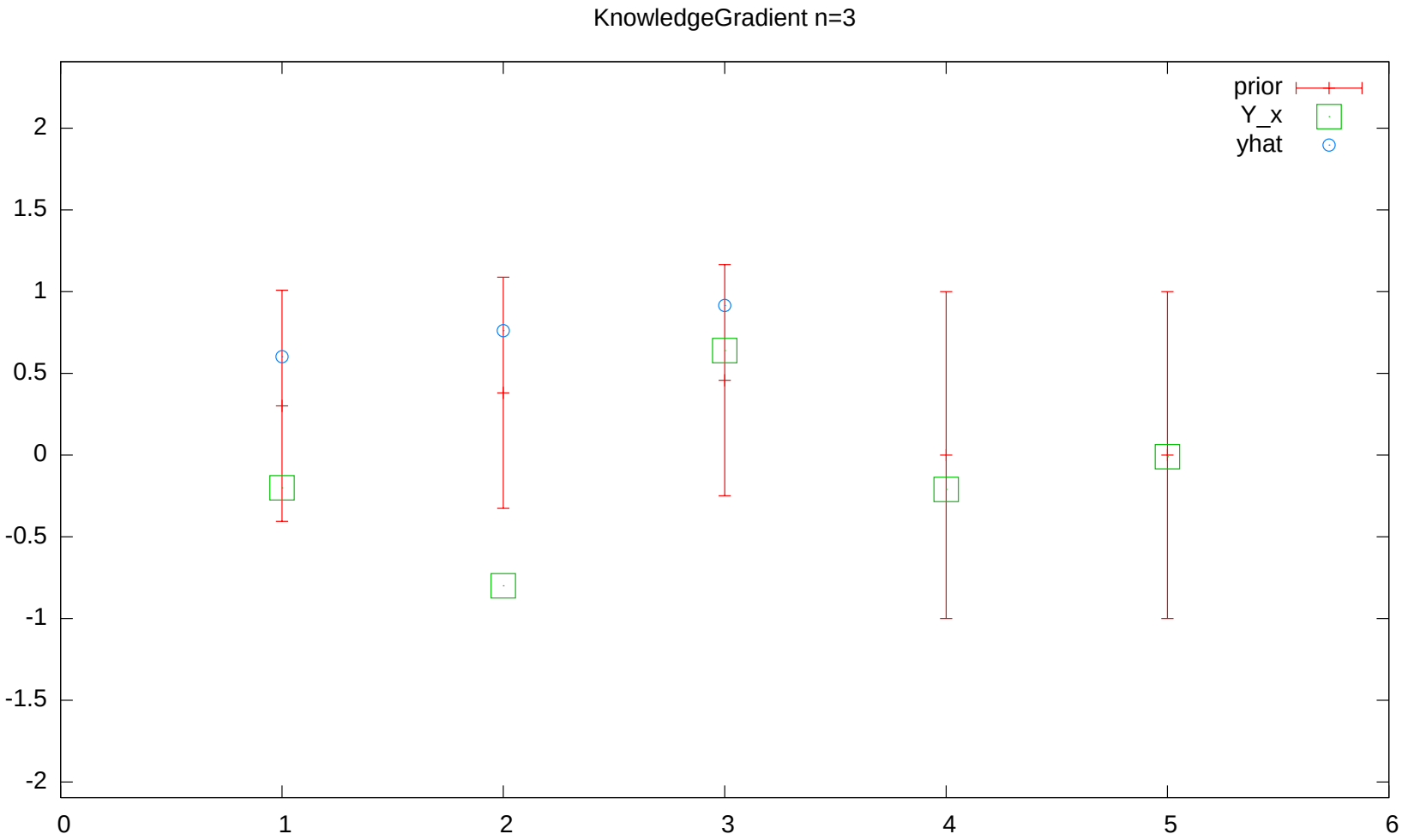
Animation of the KG method



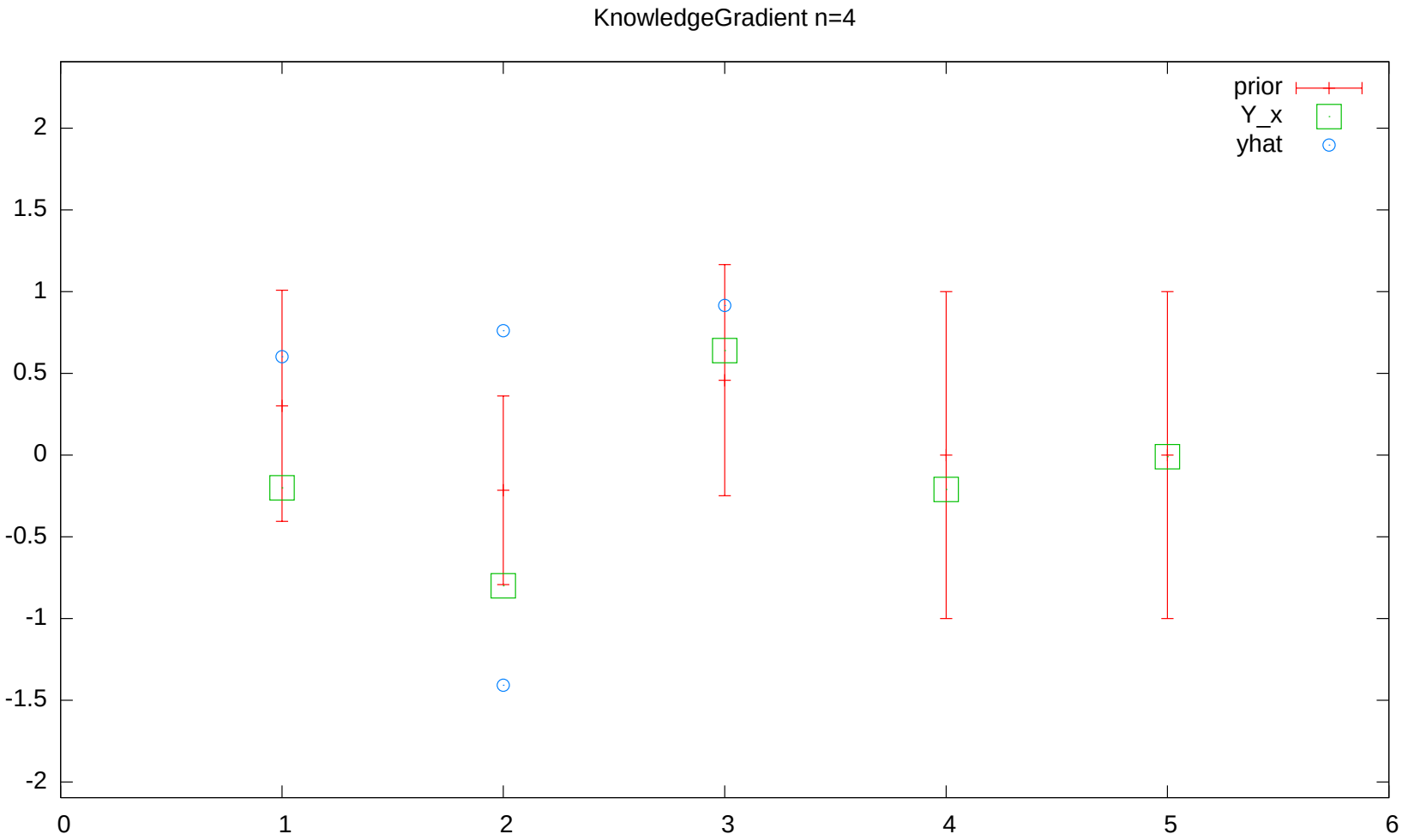
Animation of the KG method



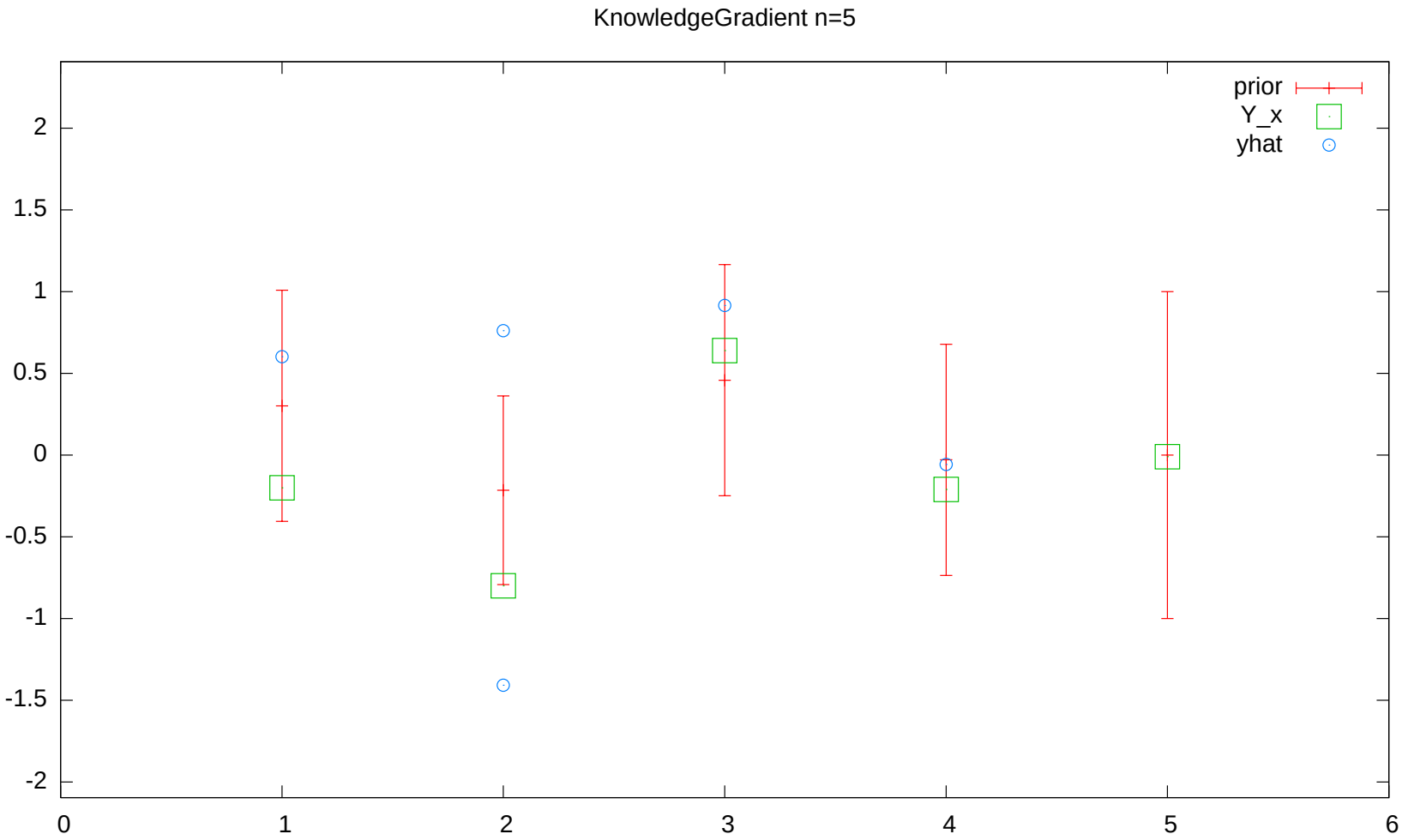
Animation of the KG method



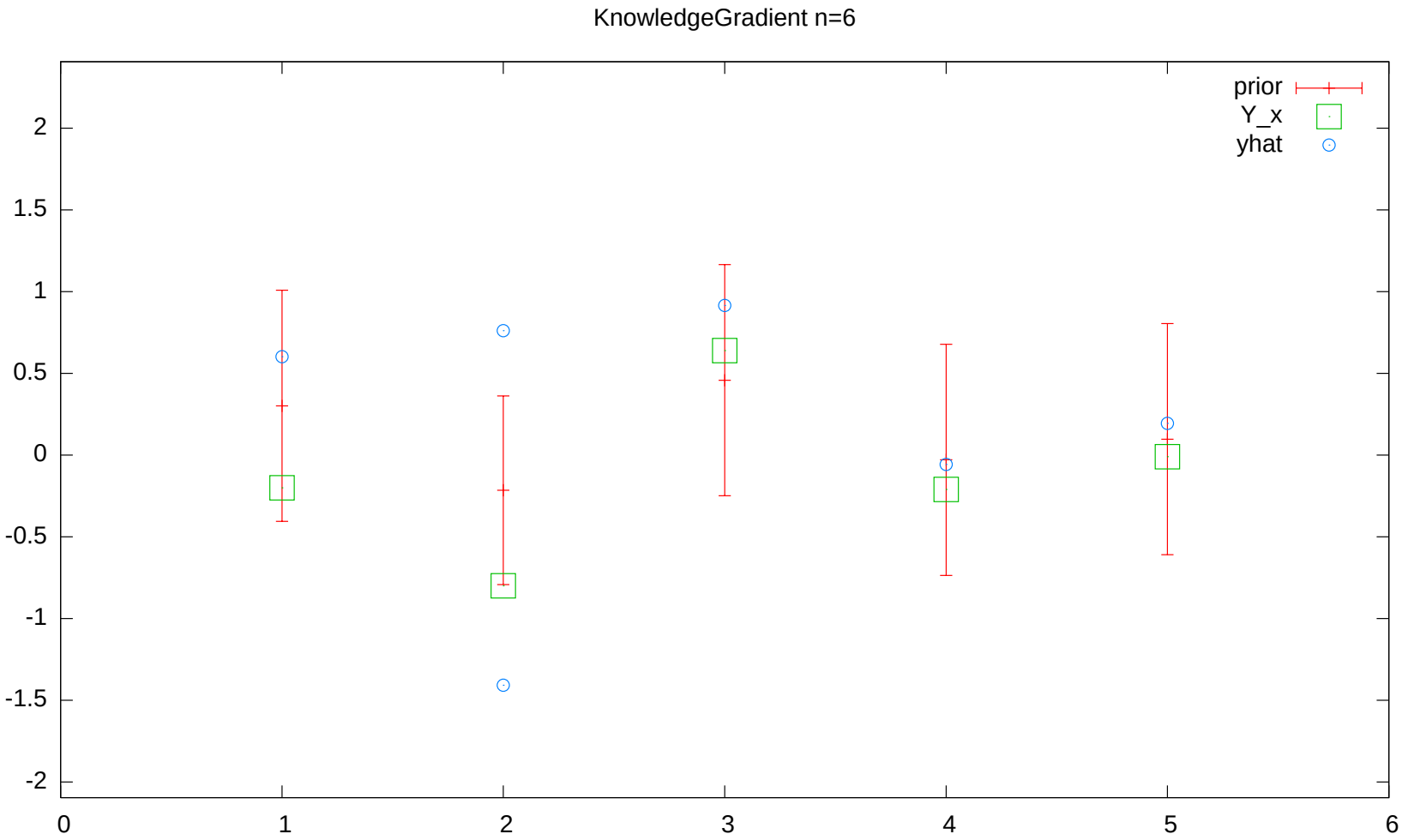
Animation of the KG method



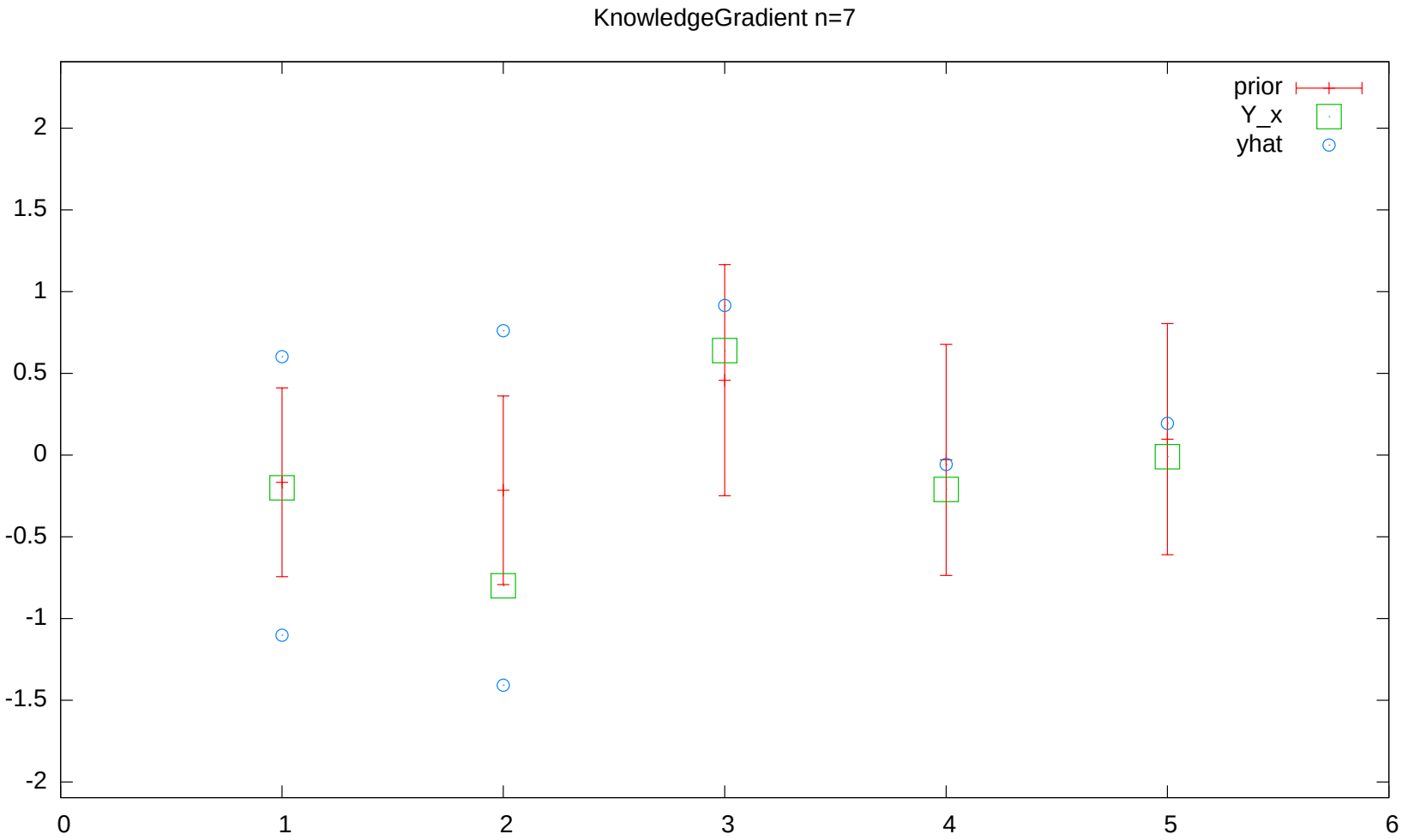
Animation of the KG method



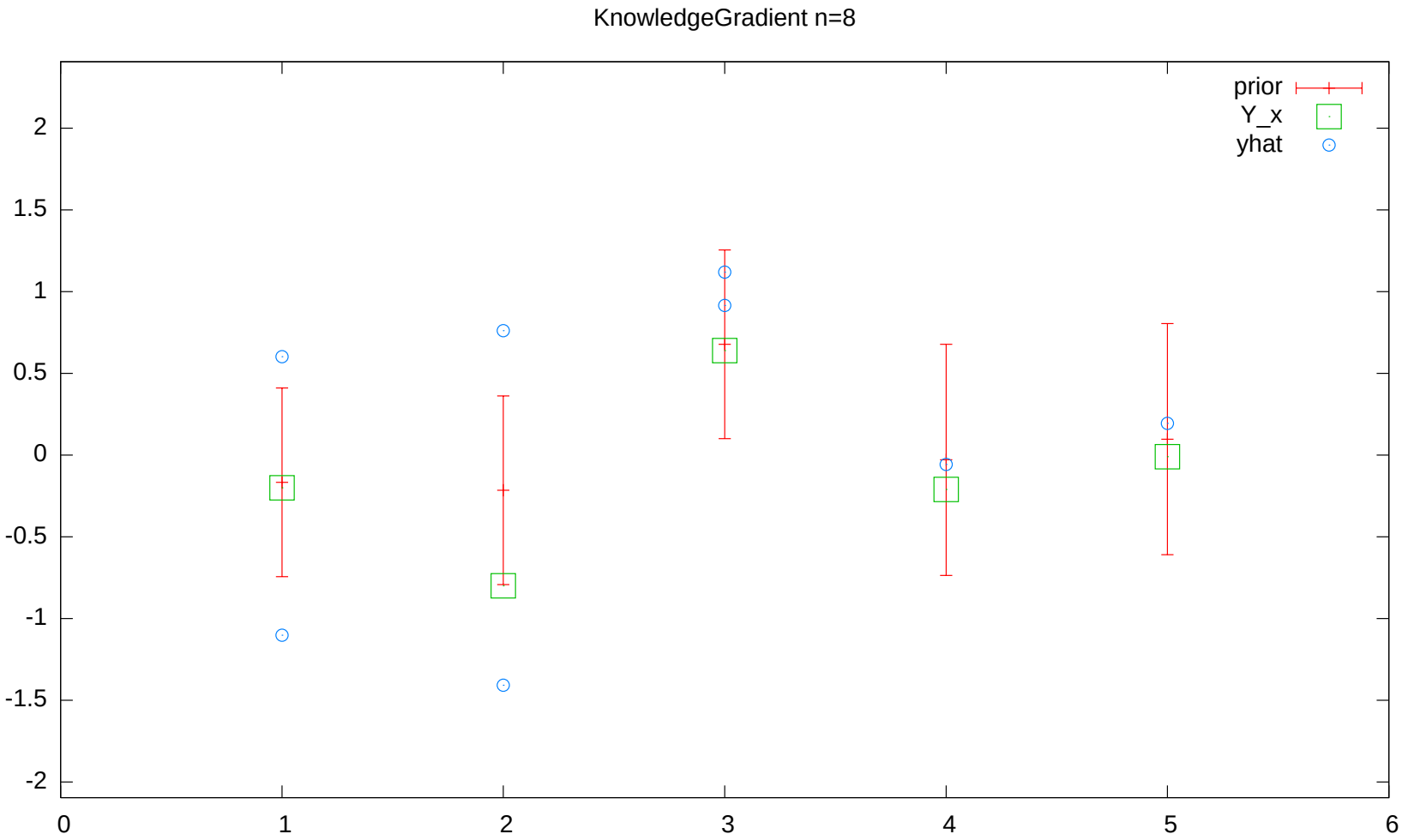
Animation of the KG method



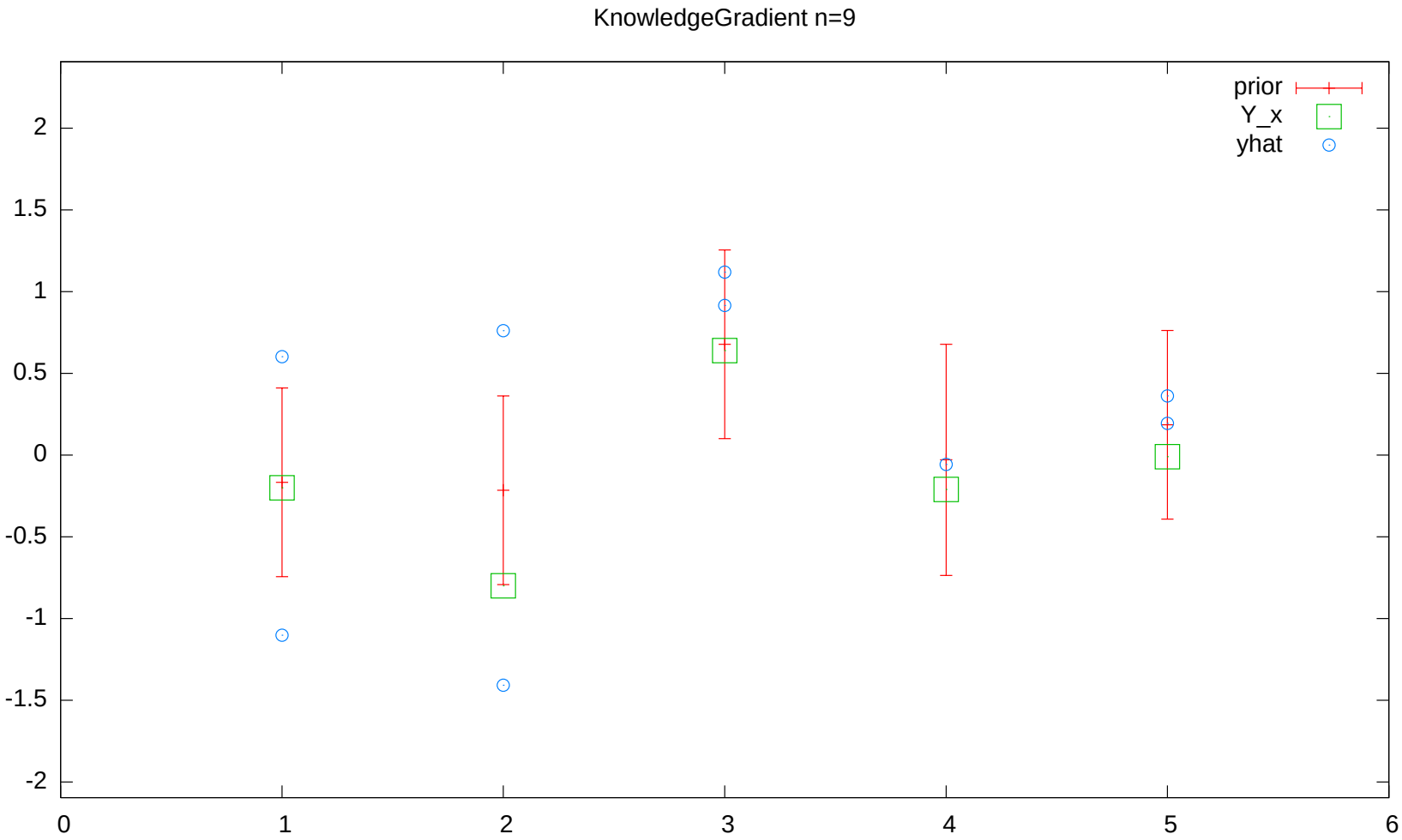
Animation of the KG method



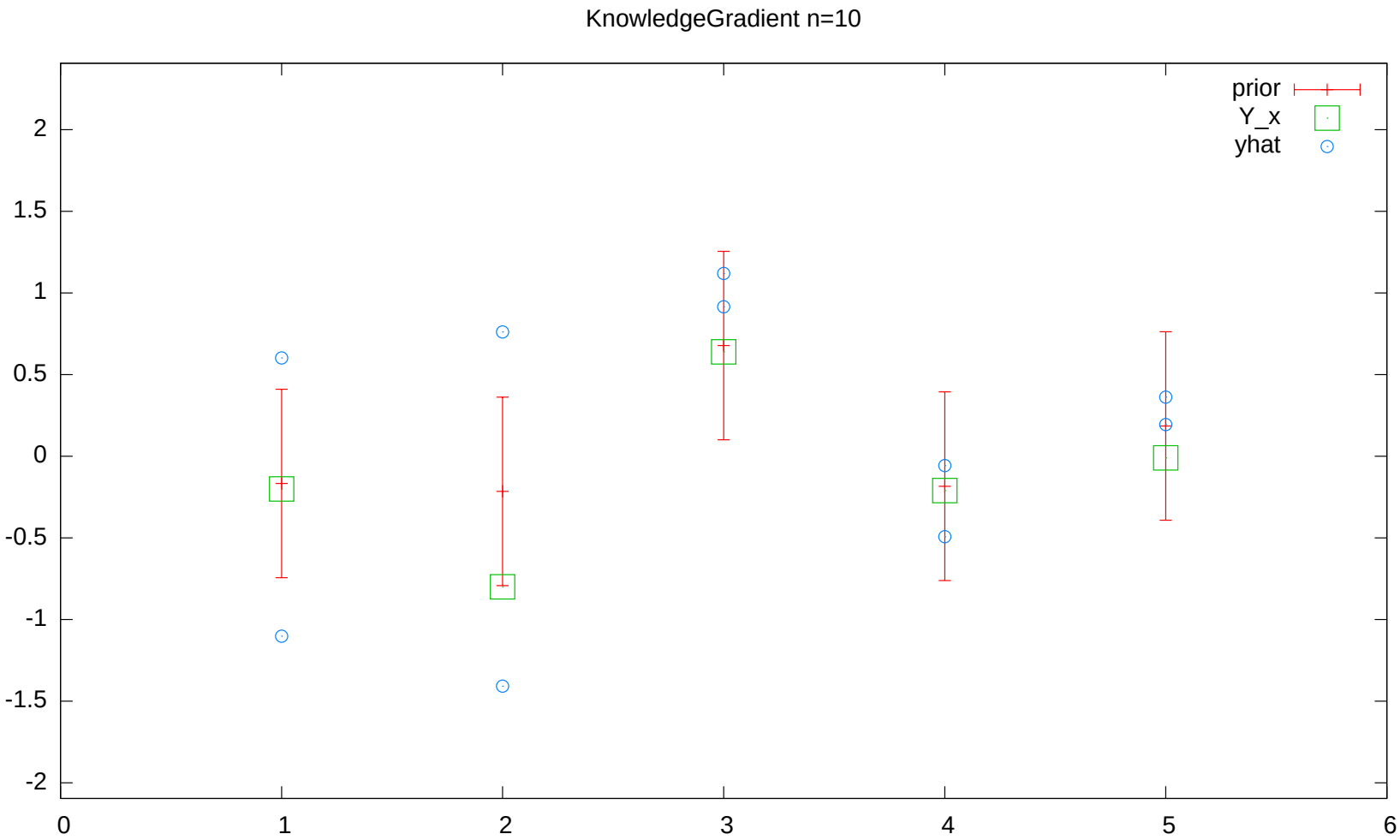
Animation of the KG method



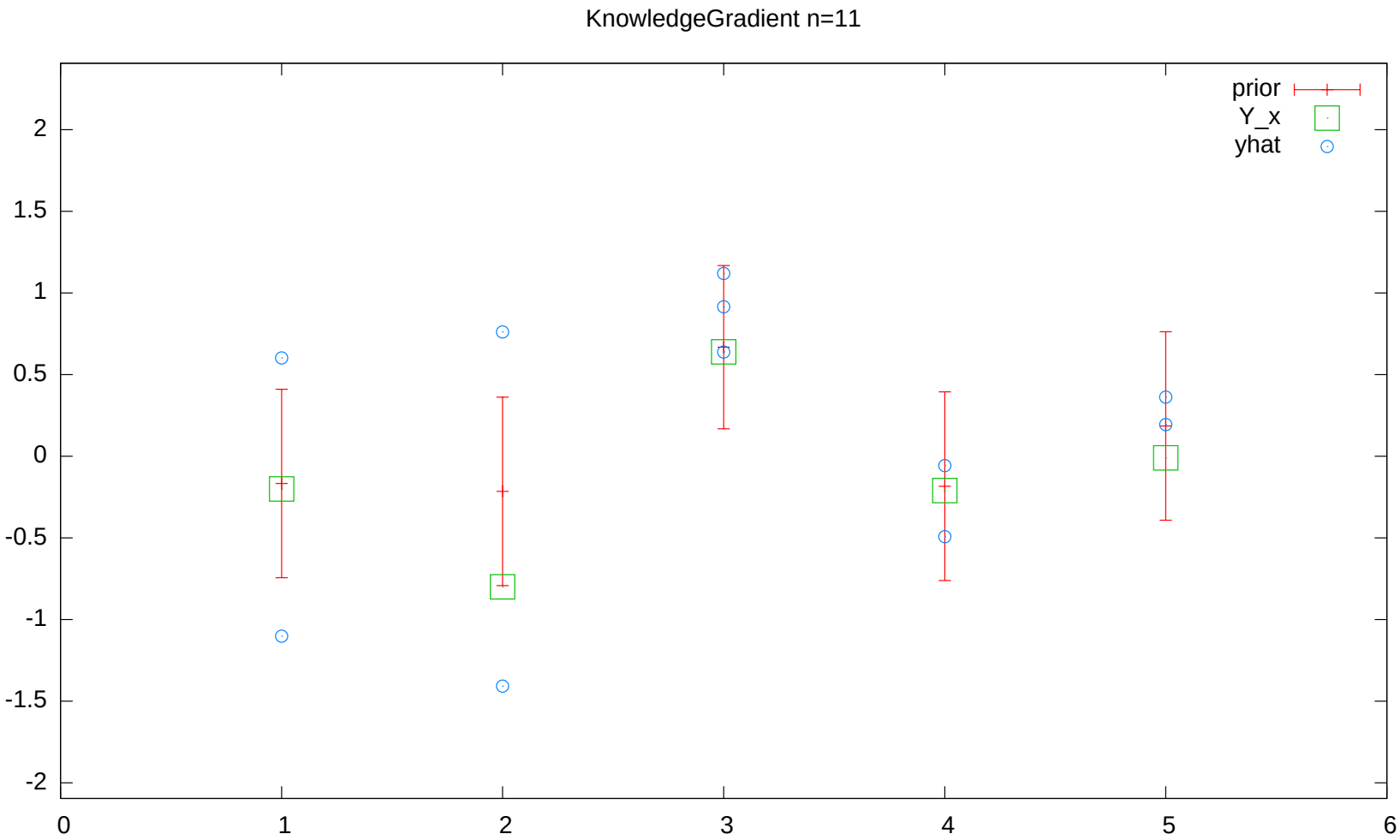
Animation of the KG method



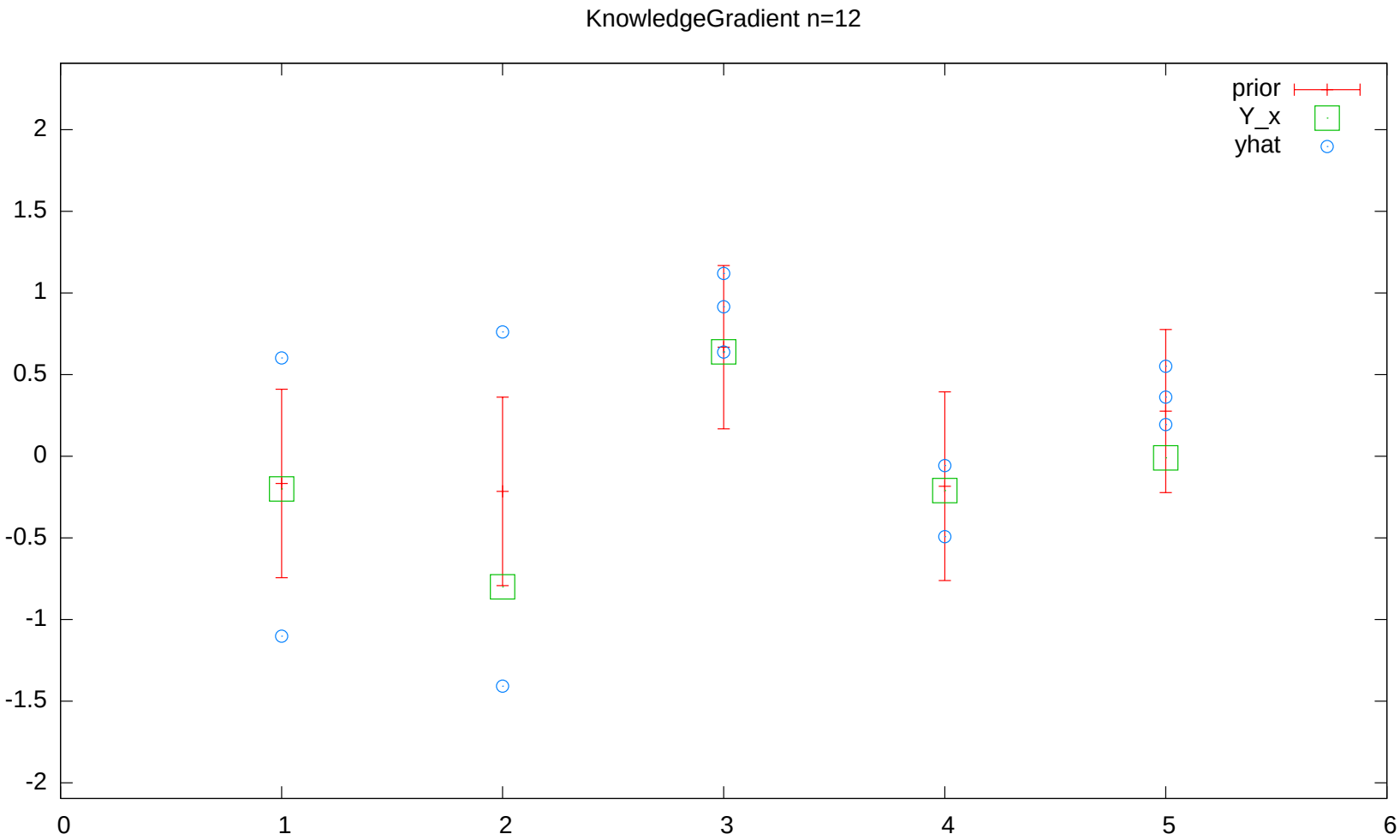
Animation of the KG method



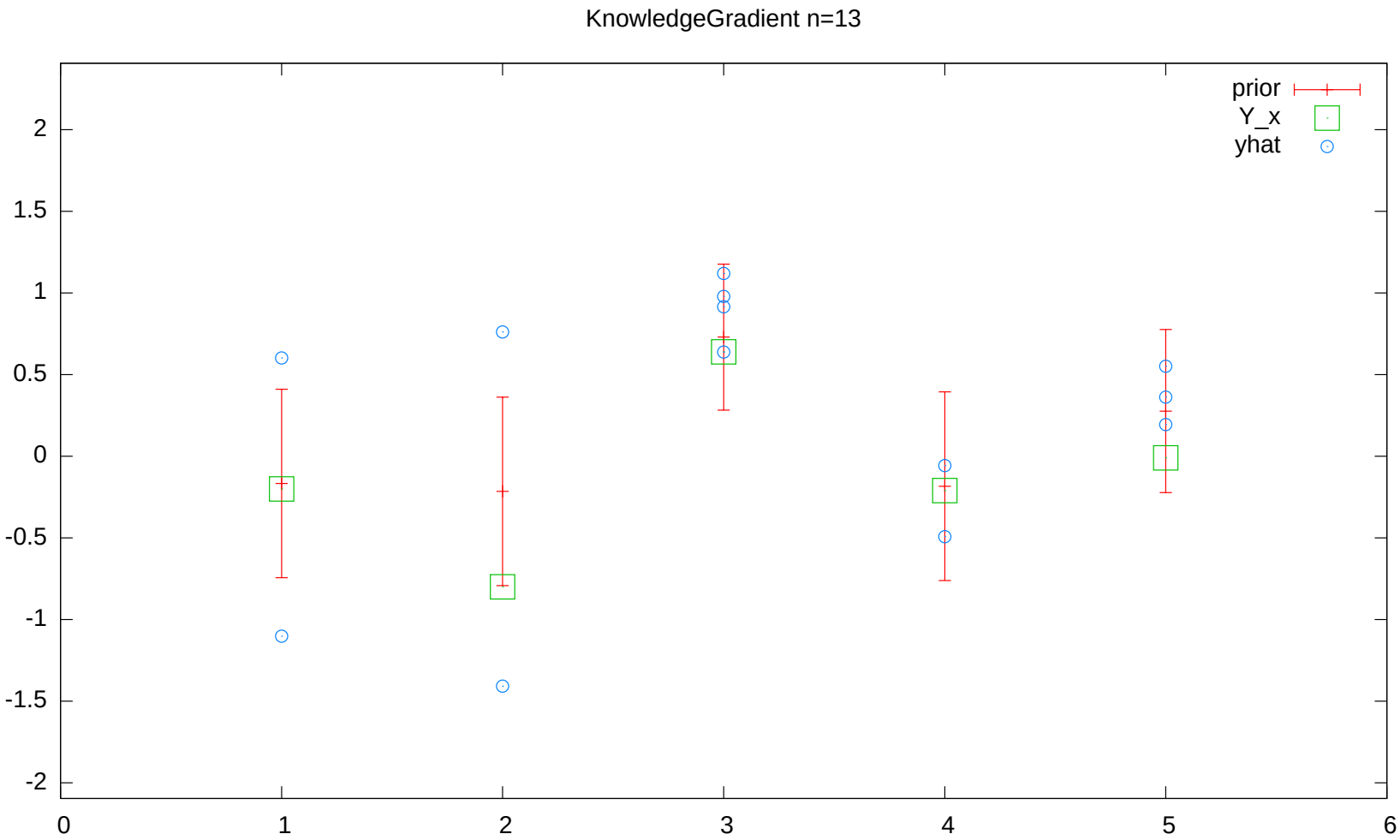
Animation of the KG method



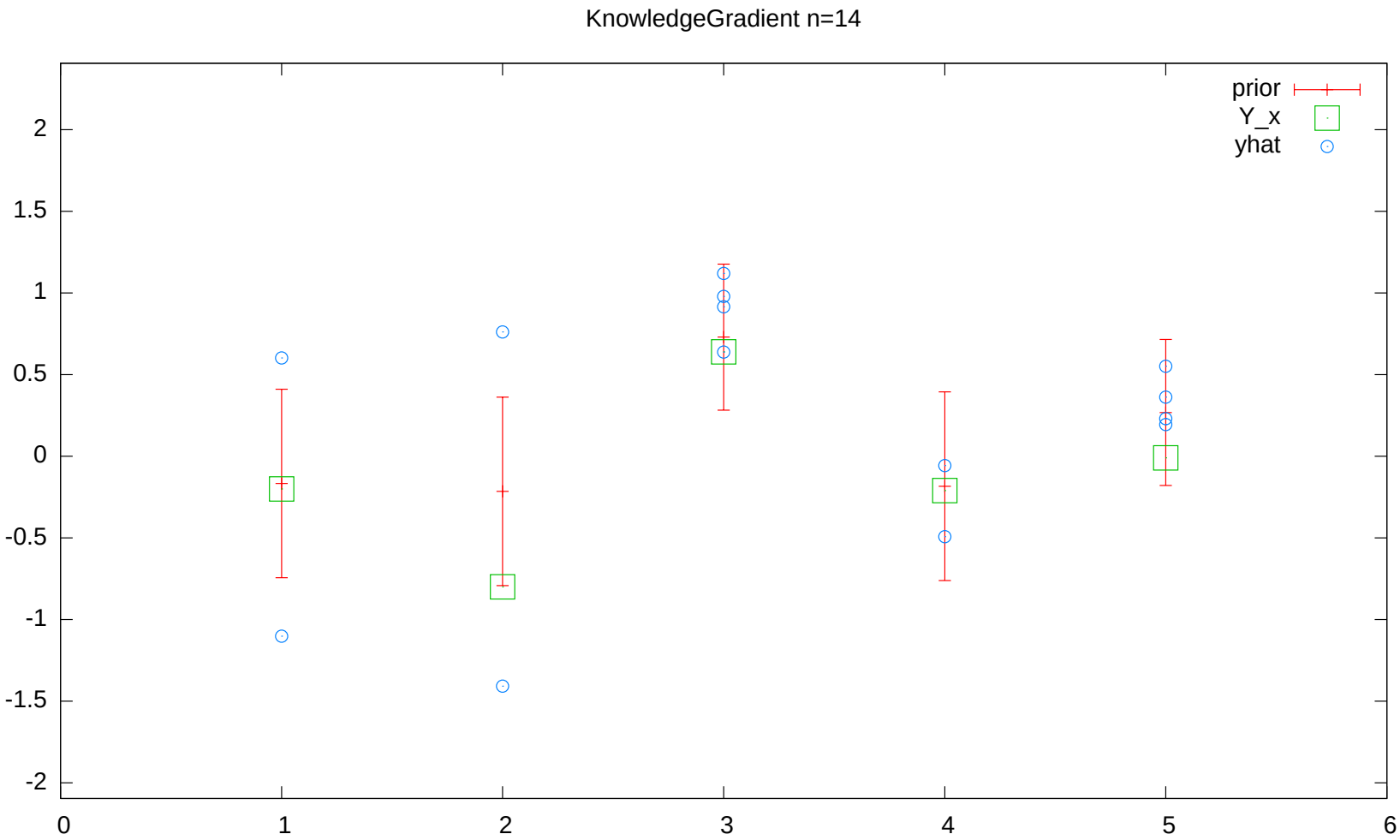
Animation of the KG method



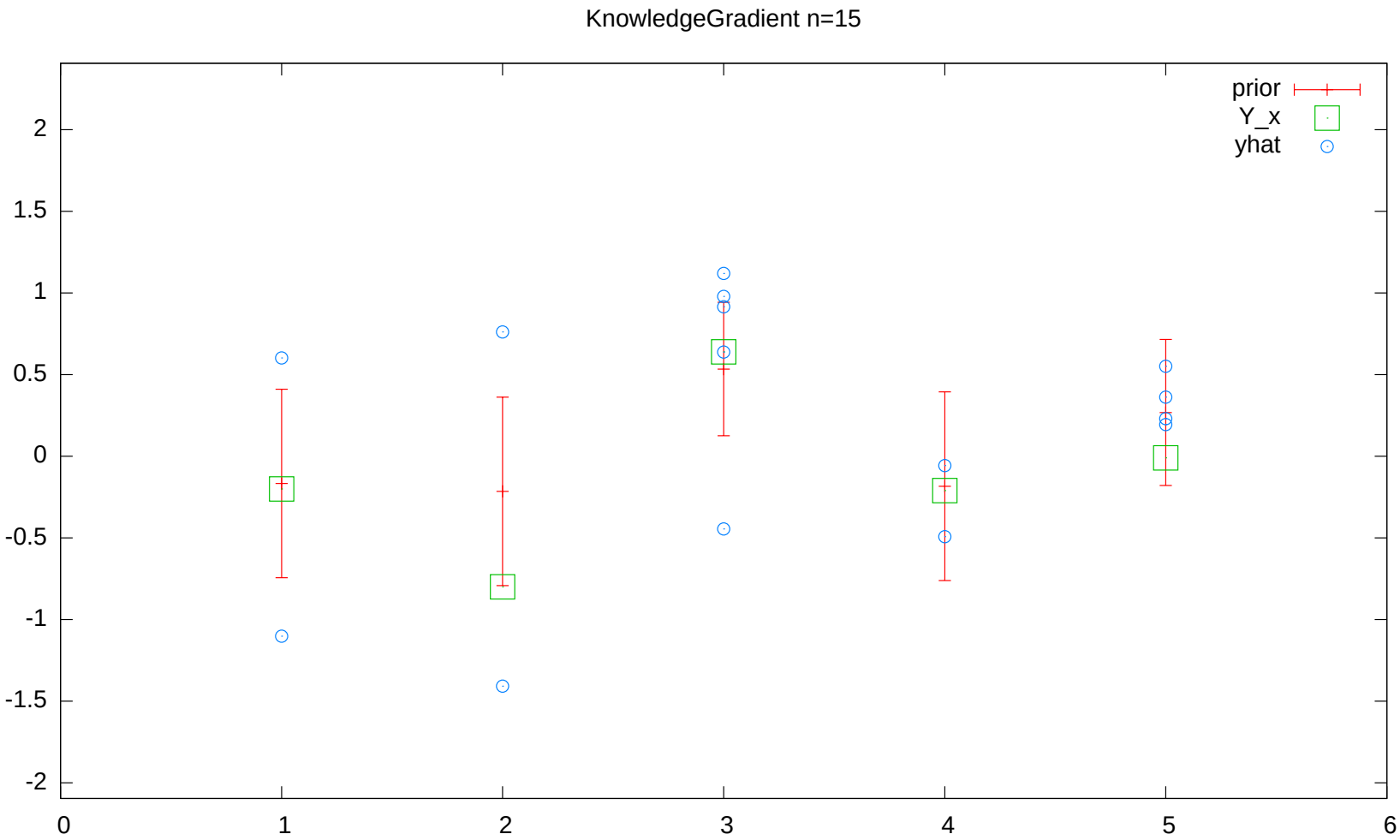
Animation of the KG method



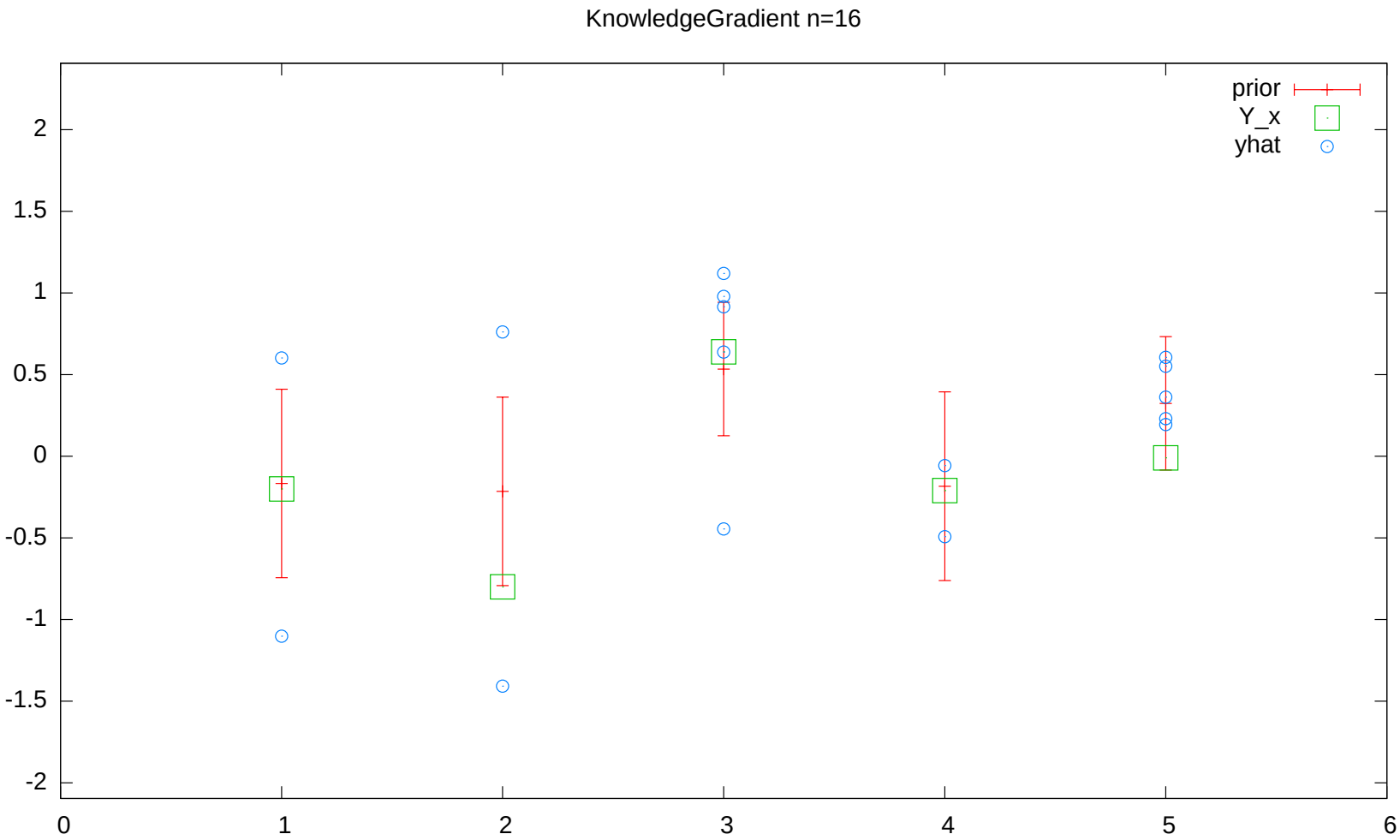
Animation of the KG method



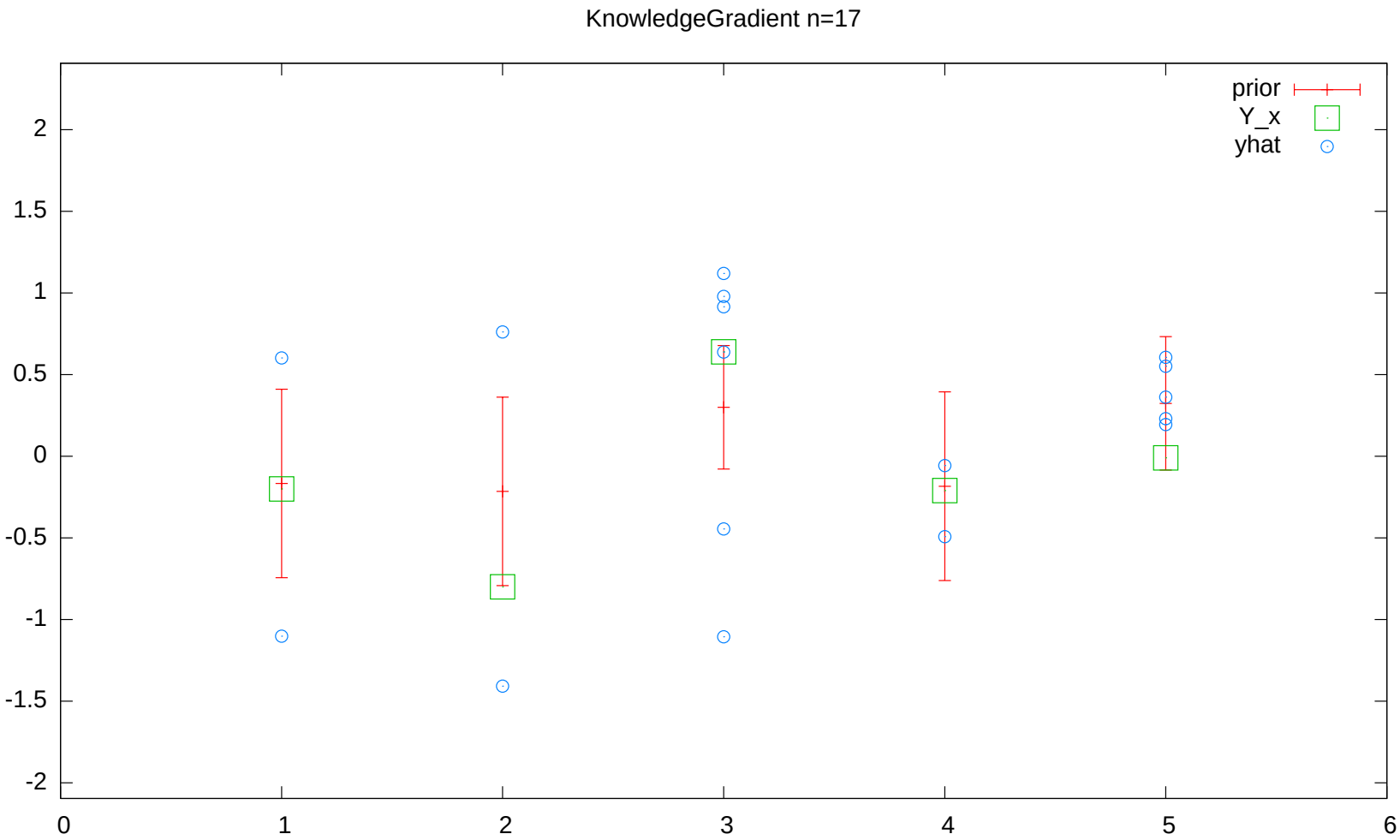
Animation of the KG method



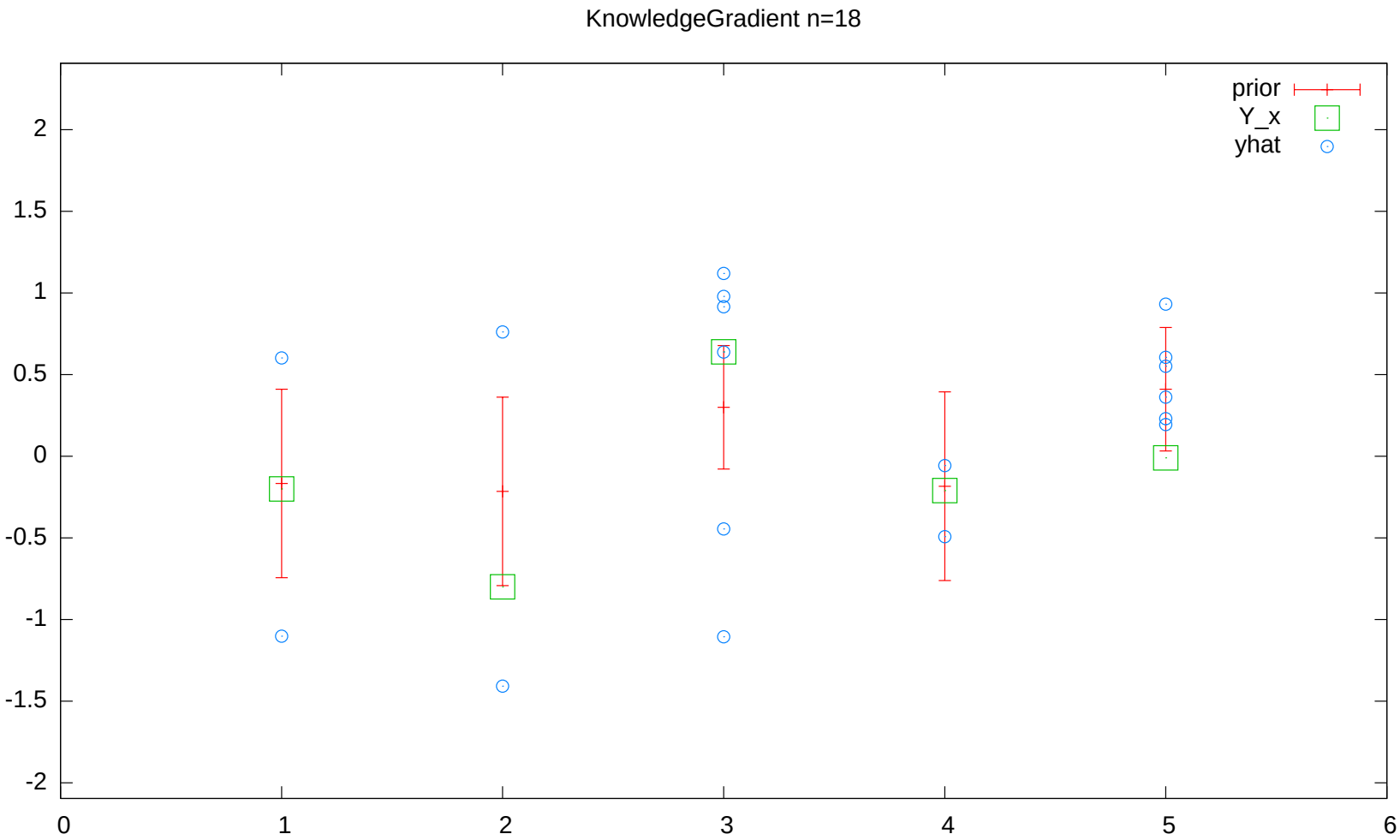
Animation of the KG method



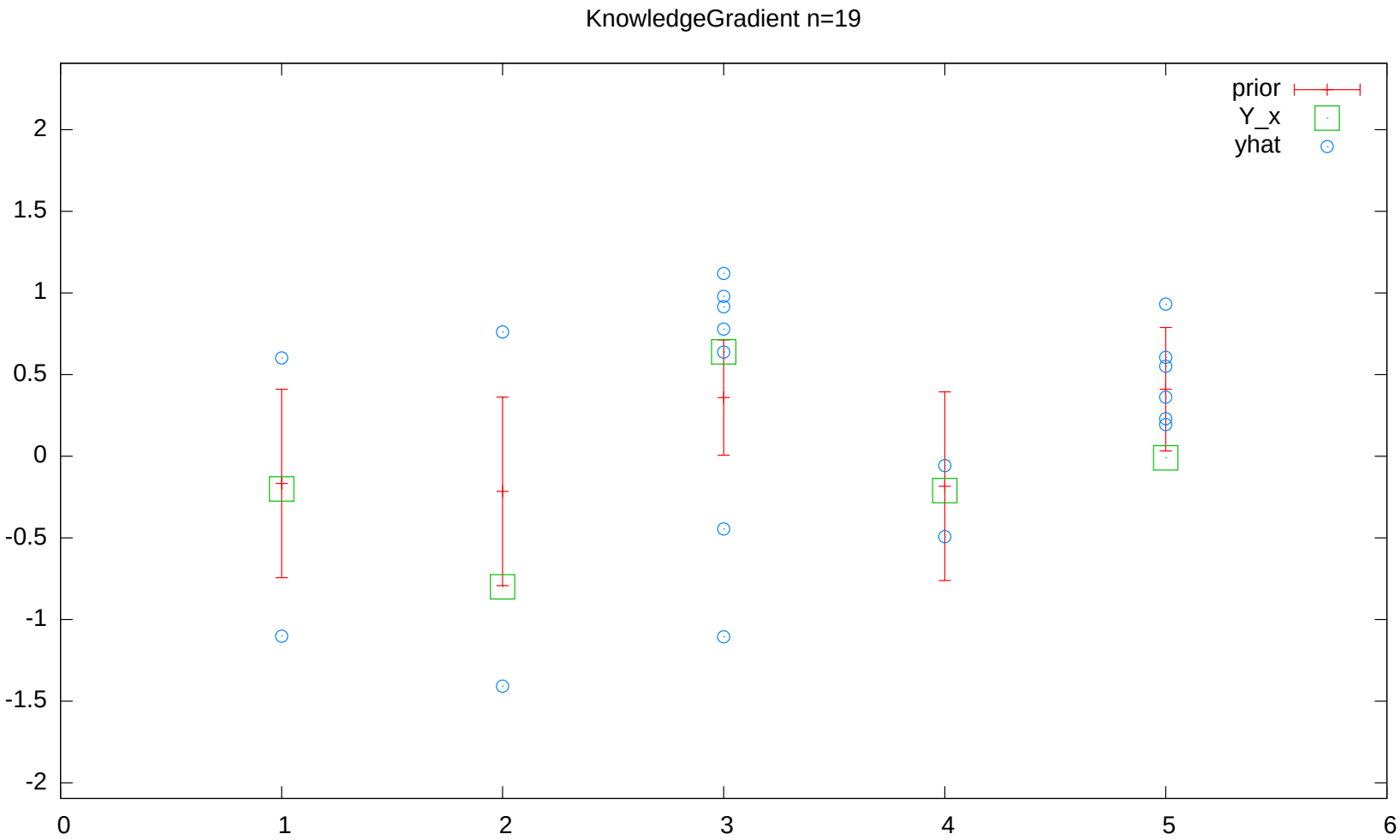
Animation of the KG method



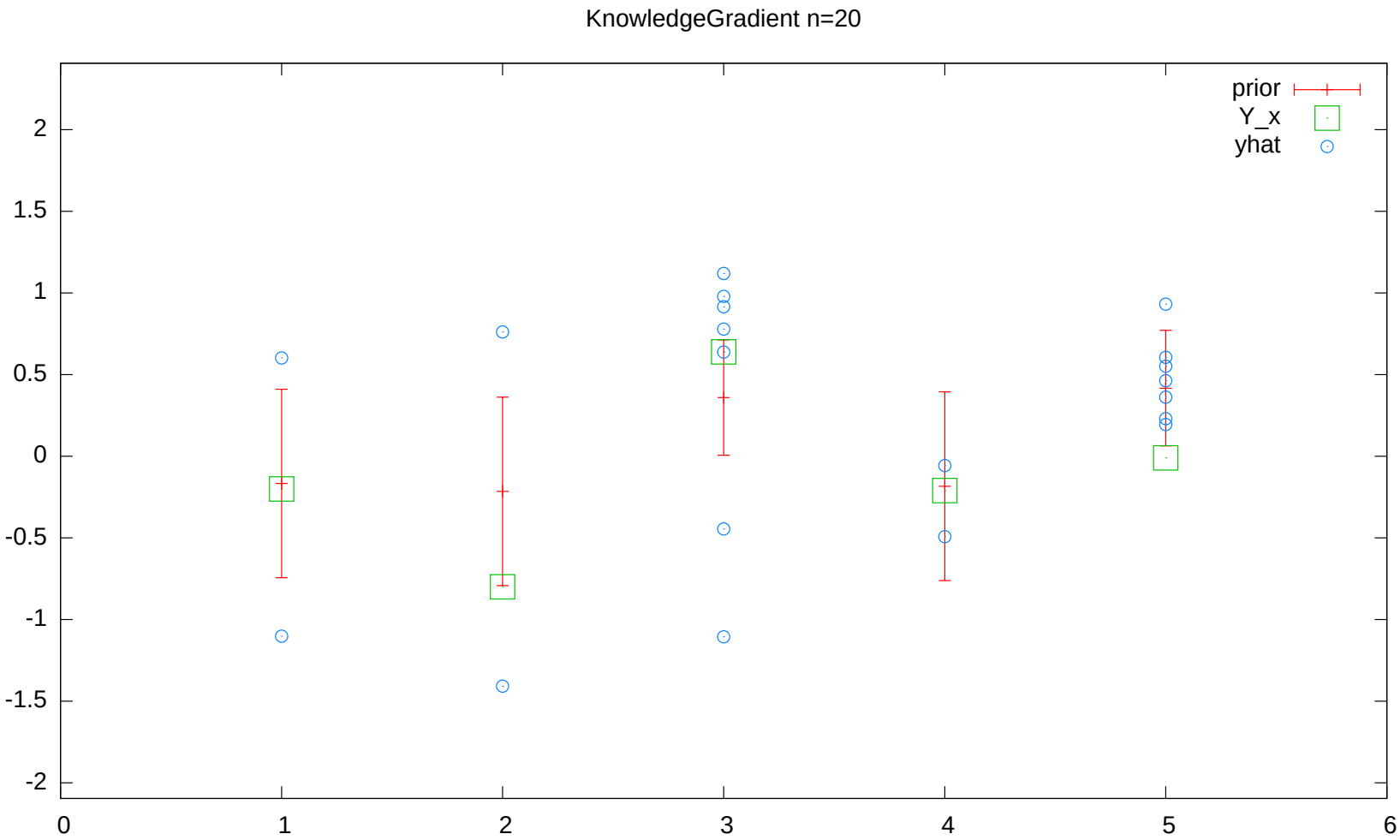
Animation of the KG method



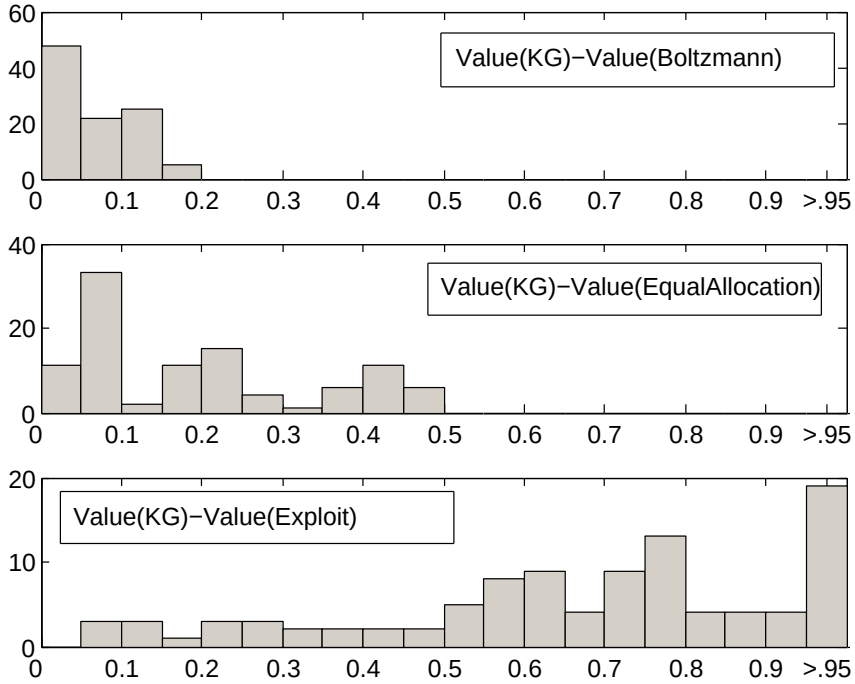
Animation of the KG method

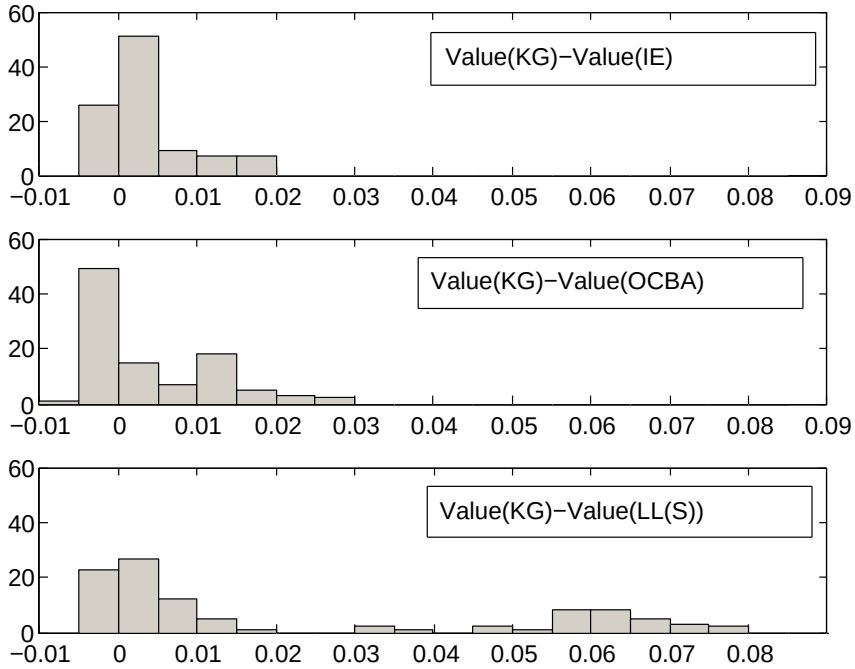


Animation of the KG method



The KG method works well





Histogram of the sampled difference in value for competing policies aggregated across the 100 randomly generated problems.

Outline

The knowledge-gradient method is good, but it not optimal in general

- The KG method works well against other algorithms proposed for this problem.
- The KG method is optimal if we have only sample remaining.
- But in general, multiple samples remain.
- What is the best algorithm in general?

The optimal algorithm is the solution to a dynamic program

- The conditional expected value we receive, given what we know at time N, is $\max_x \mu_{N,x}$.
- Define $V_N = V_N(\mu_N, \sigma_N^2) = \max_x \mu_{N,x}$.
- At time $N-1$, the optimal choice of x_N is the one that maximizes the expected value of this reward,

$$\arg\max_{x_N} \mathbb{E}_N[V_N|x_N],$$

  and this maximal expected value is

$$V_{N-1} = V_{N-1}(\mu_{N-1}, \sigma_{N-1}^2) = \max_{x_N} \mathbb{E}_{N-1}[V_N|x_N].$$

- Notation: E_n means the conditional expectation with respect to μ_n and σ_n^2; $\mu_N = (\mu_N, x : x = 1, \ldots, k)$ and similarly for σ_N^2.

In principle, we can repeat this to find the optimal rule for every x_n

We iterate backward $n = N, N-1, N-2, \ldots, 1$, where in each stage n:

- We computed $V_{n+1}(\mu_{n+1}, \sigma^2_{n+1})$ in the previous stage.
- The optimal choice for x_{n+1} is

$$x_{n+1} \in \arg\max_{x_{n+1}} \mathbb{E}_n[V_{n+1}(\mu_{n+1}, \sigma^2_{n+1})|x_{n+1}]$$

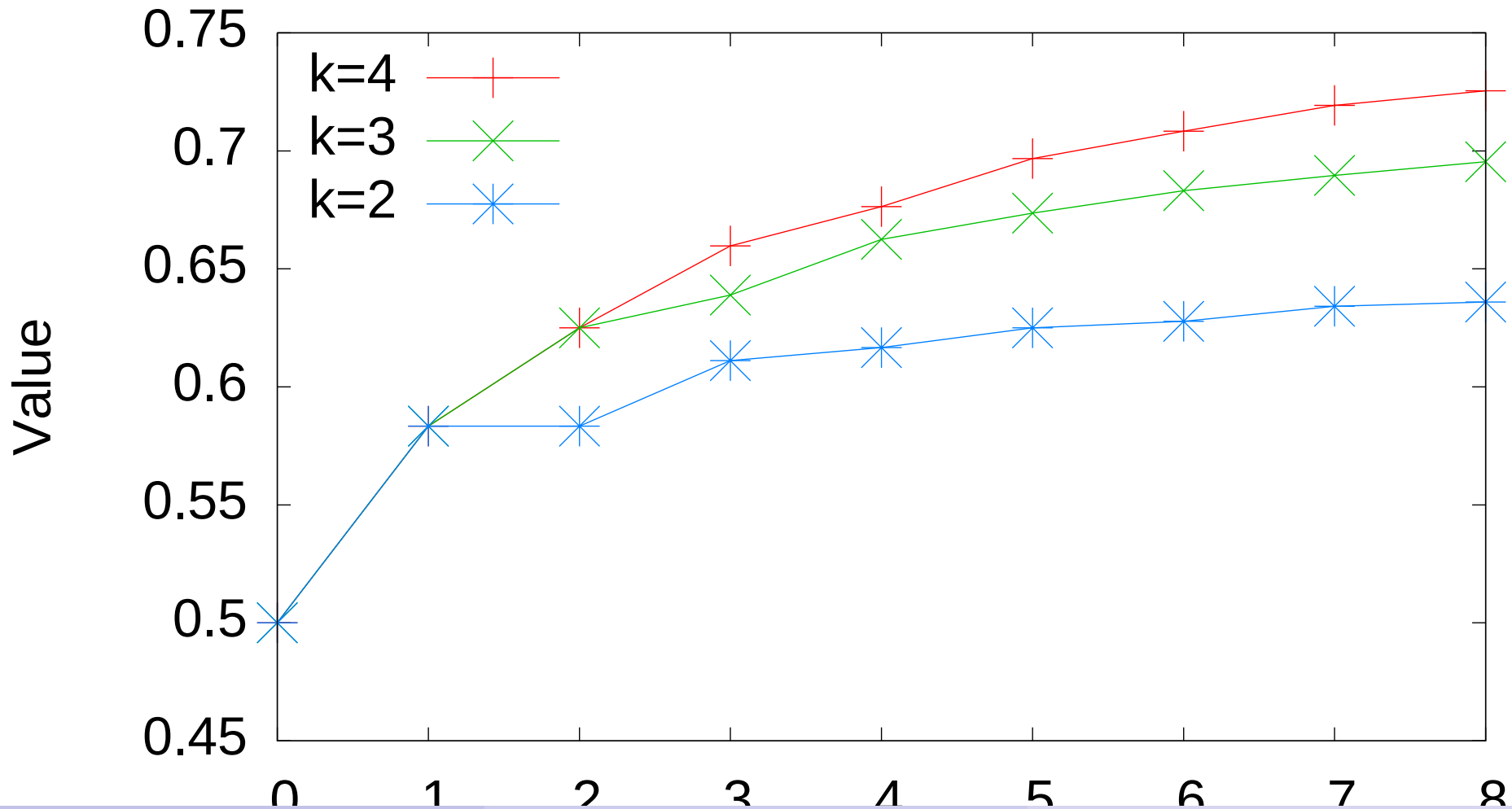
- The value of this decision is

$$V_n(\mu_n, \sigma_n) = \max_{x_{n+1}} \mathbb{E}_n[V_{n+1}(\mu_{n+1}, \sigma^2_{n+1})|x_{n+1}].$$

This is **dynamic programming**.

We can solve the DP exactly for small problems

Here is the value function for a Bayesian ranking and selection problem with Bernoulli (0/1) observations, and independent beta prior distributions.



For large problems, this does not work because of the curse of dimensionality

- To use dynamic programming, we need to compute and store $V_n(\mu_n, \sigma_n^2)$ for each possible value of μ_n and σ_n^2. (We need to compute V_n for every n, but at any given time we only need V_n and V_{n+1} in memory.)
- There are infinitely many possible values for μ_n. We can discretize, but it is a vector in k dimensions, and so discretizing into m pieces in each dimension allows for m^k possible values.
- σ_n^2 only takes finitely many values, since $(\sigma_{nx}^2)^{-1}/\lambda^{-2}$ is the number of samples of alternative x, but there are still $k^n/n!$ possible values.
- For large values of k (say, $k > 10$), solving the dynamic program is computationally intractable.
- For such large values of k, we recommend using the KG policy.

The KG method has nice optimality properties

The dynamic programming equations to prove certain optimality properties of the KG policy:

- The knowledge-gradient policy is optimal when $N = 1$.
- The knowledge-gradient policy is asymptotically optimal as $N \to \infty$.
- For other N, the knowledge-gradient policy's suboptimality is bounded by

$$V^{KG,n}(S^n) \geq V^n(S^n) - \frac{N-n-1}{\sqrt{2\pi}} \max_x \tilde{\sigma}_x^n,$$

where $V^{KG,n}$ gives the value of the knowledge-gradient policy and V^n the value of the optimal policy, both with $N-n$ measurements remaining.

The KG method has nice optimality properties

If there are exactly 2 alternatives (M=2), the knowledge-gradient policy is optimal. In this case, the optimal policy reduces to

$$x^n = \arg\max_x \sigma_x^n.$$

The KG method has nice optimality properties

If there is no measurement noise, and alternatives may be reordered so that

$$\mu_1^0 \geq \mu_2^0 \geq \ldots \geq \mu_M^0$$
$$\sigma_1^0 \geq \sigma_2^0 \geq \ldots \geq \sigma_M^0,$$

then the knowledge-gradient policy is optimal.

Outline

Conclusion

- We gave an introduction to Bayesian ranking and selection, which is one of many optimal learning problems.
- We showed how Bayesian statistics and a one-step optimality analysis can be used to derive the KG policy for this problem.
- In the seminar today, we will look at another optimal learning problem: simulation optimization, with correlated Bayesian prior distributions.
- Knowledge-gradient methods offer a convenient yet principaled way to develop algorithms for a wide variety of optimal learning problems.

For further reading

- P.I. Frazier, "Tutorial: Optimization via Simulation with Bayesian Statistics and Dynamic Programming," Winter Simulation Conference, 2012. (available on my website)
- W.B. Powell & I.O. Ryzhov "Optimal Learning", 2012. (textbook)
- The original paper on the KG method: P.I. Frazier, W.B. Powell, and S. Dayanik "A Knowledge-Gradient Policy for Sequential Information Collection," SIAM Journal on Control and Optimization, 2008.
- Other introductory materials available on my website, `http://people.orie.cornell.edu/pfrazier/`