Practical Bayesian Tomography

1509.03770

Christopher Granade, Joshua Combes and D. G. Cory

videos, tutorials and source code
→ www.cgranade.com/research/pbt *or tap*

Introduction

We want tomographic methods that:

- Provide accurate estimates
- Characterize their uncertainty
- Include prior information
- Track time-dependence
- Are "off-the-shelf"

Bayes' rule gives the first two. → doi:10/cn772j
Can be done numerically. → doi:10/s86, doi:10/7nt
We do the rest.

Prior Information

Big idea: use ensemble of amplitude damping channels to encode prior assumptions into uninformative priors.

- Easy to sample
- Inherits other assumptions by convexity
- Includes both state and process tomography
- Robust (doesn't contract support)
- Function *only* of desired estimate

$$\rho_{\text{sample}} = (1-\epsilon)\rho + \epsilon\rho_*$$

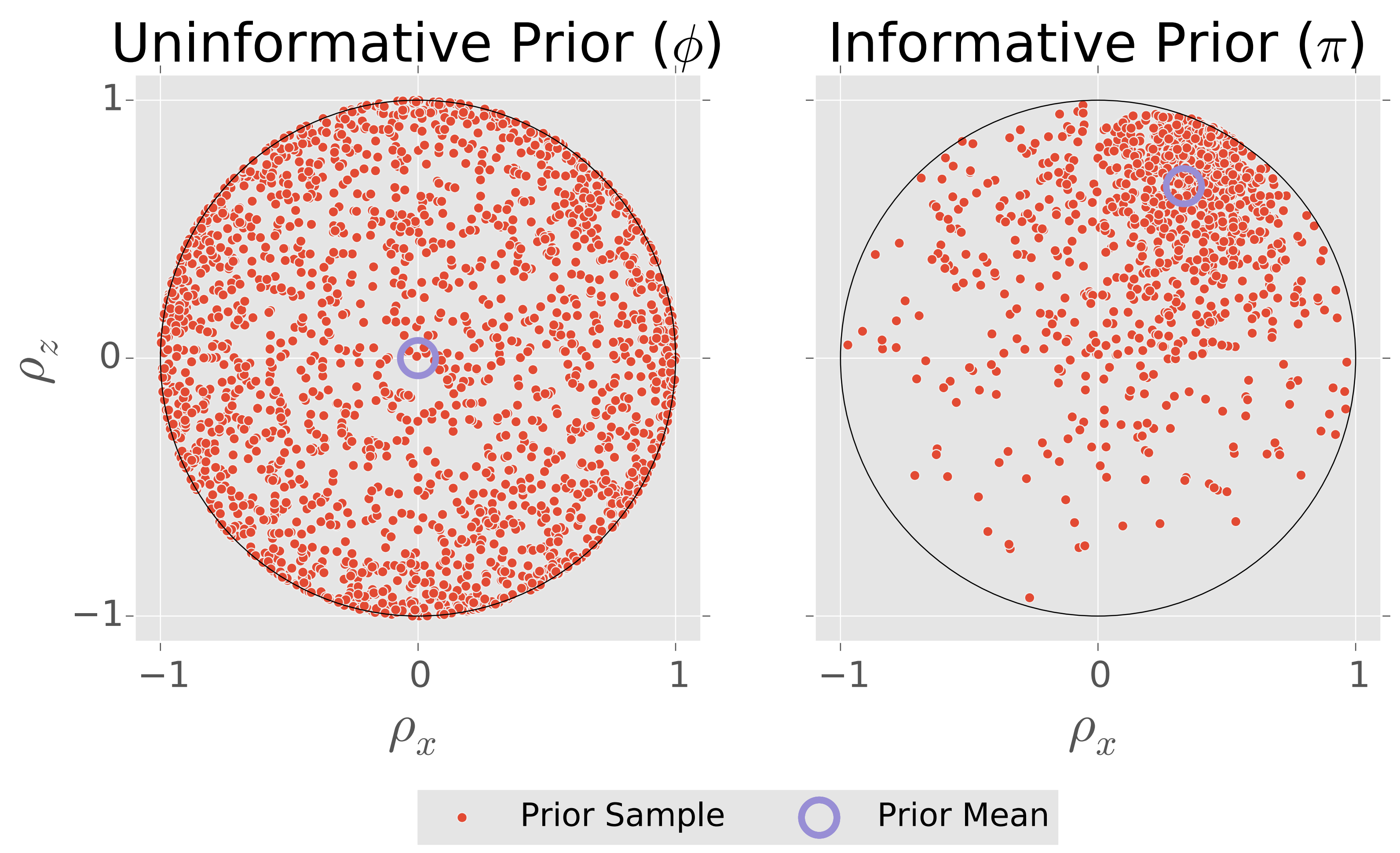
$$\rho \sim \text{uninformative prior}$$

$$\epsilon \sim \text{Beta}(\alpha, \beta)$$

$$\rho_* = \frac{\alpha+\beta}{\alpha}\left(\rho_\mu - \frac{\beta}{\alpha+\beta}\frac{\mathbb{1}}{D}\right)$$

$$\alpha = 1$$

$$\beta = \frac{D\lambda_{\min}}{1 - D\lambda_{\min}}$$



Samples of a rebit mixed state, drawn both from an uninformative (Ginibre) prior, and from our informative prior.

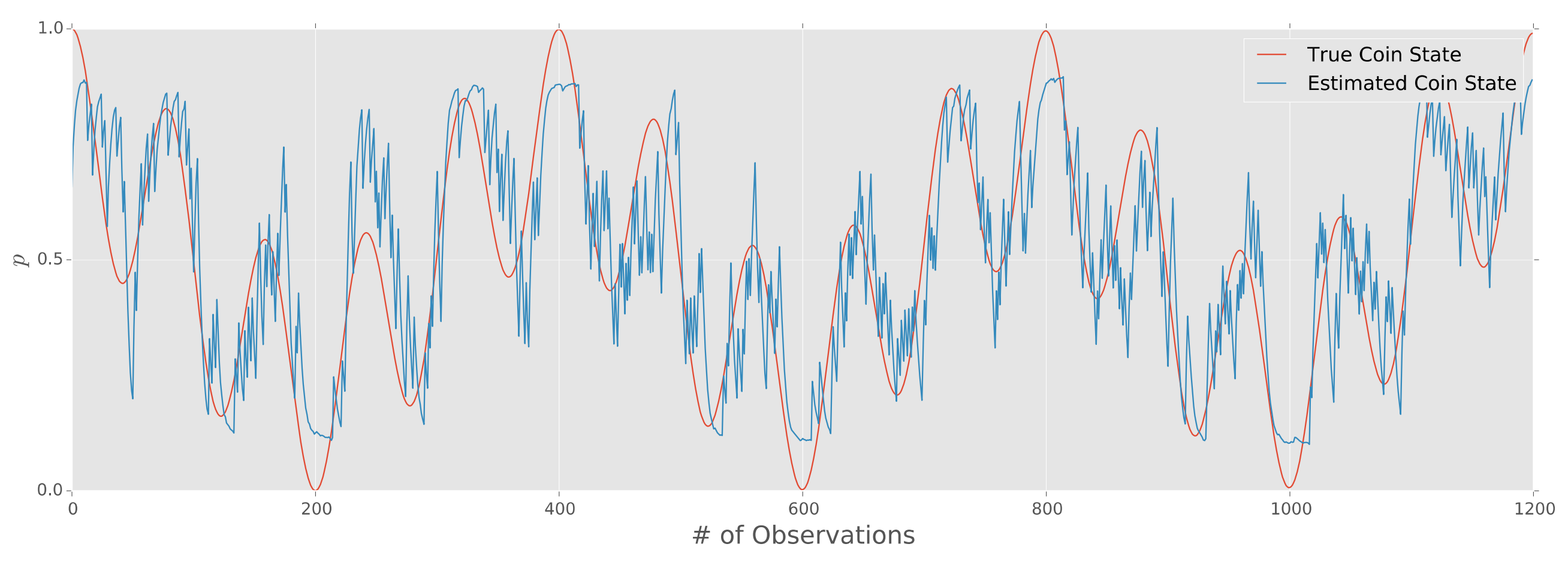
Algorithm and Software

We use particle filtering to update. → doi:10/s87
We implement in *QInfer*. → doi:10/bchq

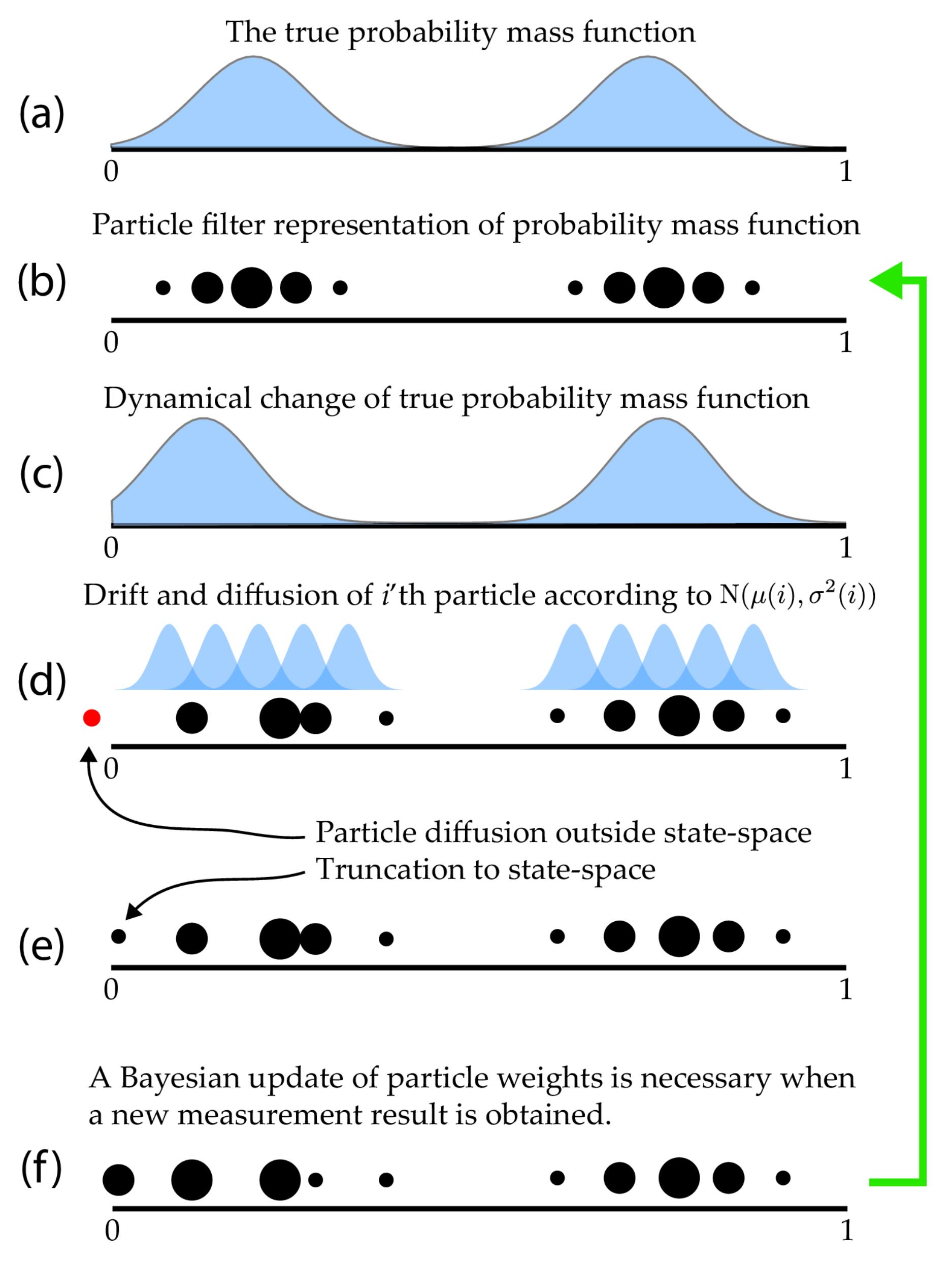
```
from __future__ import division
import qinfer as qi
import qutip as qt

I, X, Y, Z = qt.qeye(2), qt.sigmax(), qt.sigmay(), qt.sigmaz()
prior_mean = (I + (2/3) * Z + (1/3) * X) / 2

basis = qi.tomography.pauli_basis(1)
fid_prior = qi.tomography.GinibreReditDistribution(basis)
prior = qi.tomography.GADFLIDistribution(fid_prior, prior_mean)
model = qi.BinomialModel(qi.tomography.TomographyModel(basis))
updater = qi.smc.SMCUpdater(model, 2000, prior)
heuristic = qi.tomography.RandomPauliHeuristic(updater,
    other_fields={'n_meas': 40})

for idx_exp in xrange(50):
    experiment = heuristic()
    # Your data goes here! ♥
    datum = model.simulate_experiment(true_state, experiment)
    updater.update(datum, experiment)
```

State Tracking

To track time-dependence, we use the CONDENSATION algorithm: → doi:10/cc76f6
Update each particle by *drift* and *diffusion* steps. We then *truncate* negative eigenvalues.



Stochastic tracking even follows deterministic evolution (worst-case).


The true probability mass function
(a)
Particle filter representation of probability mass function
(b)
Dynamical change of true probability mass function
(c)
Drift and diffusion of i'th particle according to $N(\mu(i), \sigma^2(i))$
(d)
Particle diffusion outside state-space
Truncation to state-space
(e)
A Bayesian update of particle weights is necessary when a new measurement result is obtained.
(f)
0
1

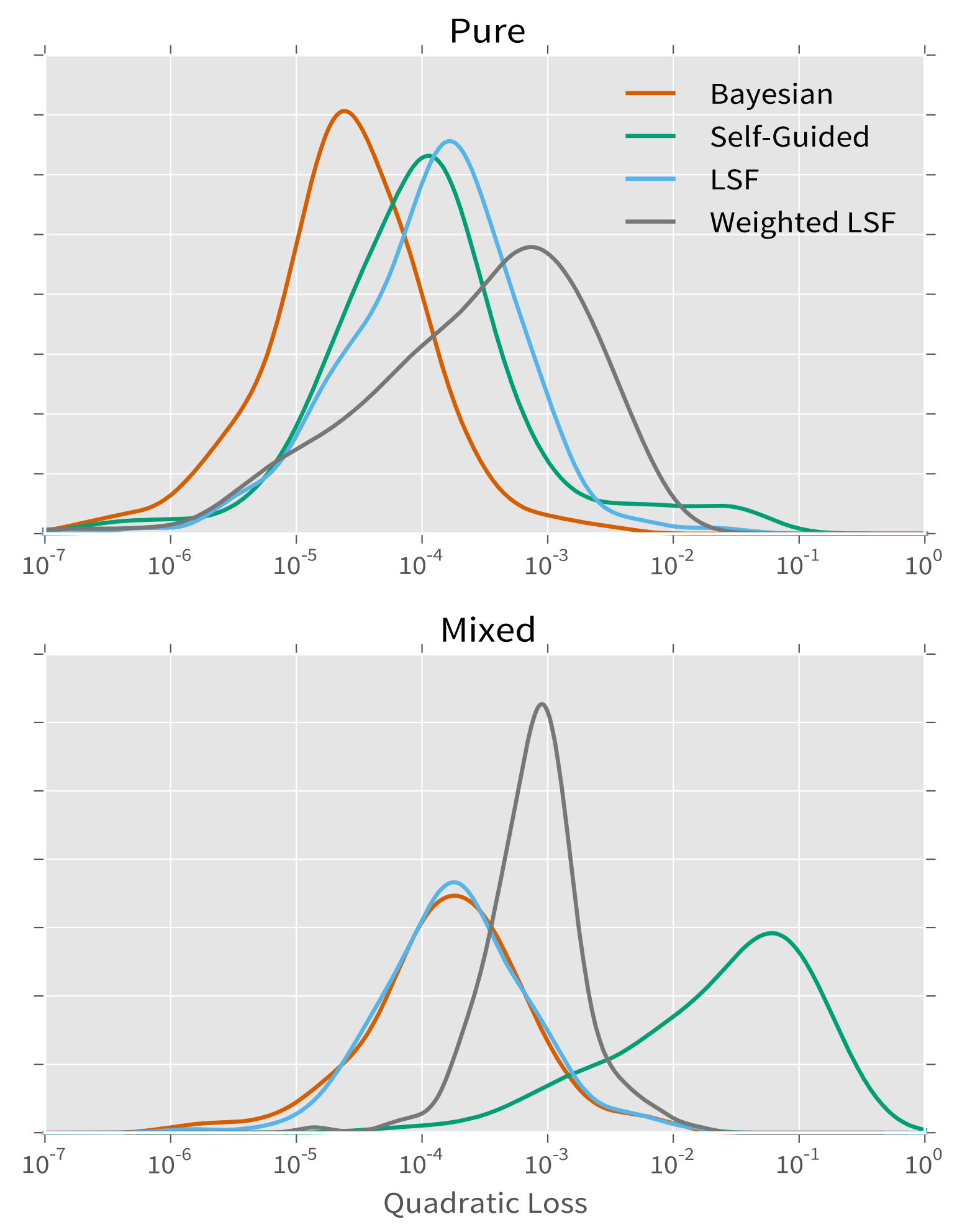
Our algorithm thus relaxes the assumption that states are static in time.

demo video → goo.gl/mkibti

Hybrid

joint work w/ Christopher Ferrie

Can also combine with *self-guided tomography* (SGT) data, giving an efficient experimental heuristic.



More on SGT: → doi:10/bchr