Numerical integration + Estimating DSGE models via filtering to recover states: Part 2

Tom Holden

http://www.tholden.org/

PhD Macroeconomics, Semester 2

Outline of today's talk

- Gaussian Quadrature/cubature
    - Gauss-Hermite
    - (Non-product) Monomial Quadrature
    - Quadrature/Cubature Kalman Filter
    - Unscented Kalman Filter

- Monte Carlo
    - Quasi-Monte Carlo
    - Markov Chain Monte Carlo (MCMC)
        - The Metropolis-Hastings Algorithm
        - Bayesian Estimation via MCMC
    - Importance sampling
        - Sequential Monte Carlo (SMC) = The Particle Filter
        - Estimating nonlinear models via the Particle Filter
        - Bayesian Estimation via SMC

Reading for today

- Canova: “Methods for applied macroeconomic research”.
  - Section 9.5 covers MCMC.
  - Section 11.4.4 covers the Particle Filter.

- Judd: “Numerical Methods in Economics”
  - Chapter 7 covers quadrature.
  - Chapter 8 covers Monte Carlo methods.
  - Chapter 9 covers Quasi-Monte Carlo.

- Stroud: “Approximate Computation of Multiple Integrals”
  - The classic work on monomial quadrature.
  - Cools & Rabinowitz (1993) contains an update.
  - And this website summarizes the key results of both works: http://nines.cs.kuleuven.be/research/ecf/mtables.html
    - A username and password is required for the tables. Ask me for them.

Numerical Integration: General Problem

- Suppose we want to evaluate:
$$\int_{x\in\mathbb{S}} g(x)\,dF(x)\,,$$
    - where $\mathbb{S}$ is some set, $g: \mathbb{S} \to \mathbb{R}$ is some function, and $F: \mathbb{S} \to [0,1]$ is the C.D.F. of some probability distribution.
    - Note that if $F(x)$ is differentiable with $F'(x) = f(x)$, this integral is $\int g(x)f(x)\,dx$.
    - In the multivariate case in which $\mathbb{S} \subseteq \mathbb{R}^d$, we need to abuse notation slightly and take $f(x) \coloneqq F'(x) \coloneqq \frac{\partial^d F(x)}{\partial x_1 \dots \partial x_d}$.
    - Throughout, lower case f will be this multiple partial derivative of F. (I.e. the p.d.f. of F.)

- With particular g and F we might be able to evaluate this integral analytically, but often in macroeconomics we won't.
- Examples include:
    - Evaluating expectations when solving a macro model globally.
    - Integrating over the current value of the state in order to get the distribution of tomorrow's state.
    - Integrating the product of the likelihood and the prior in order to get the posterior, when performing Bayesian estimation.

Quadrature via series expansion: Idea

- Suppose that we have a countable basis for the set of all integrable functions on $\mathbb{S}$. Call the basis functions $e_0, e_1, \ldots$.
- I.e., there exists $a_0, a_1, \ldots$ such that $\sum_{k=0}^{K} a_k e_k \to g$ as $K \to \infty$ (where the left hand limit is convergence under some norm on the space of functions on $\mathbb{S}$).
  - For example, we have seen that when $\mathbb{S} = [0,1]$, we can take $e_k = e^{-2\pi i x n(k)}$, where $n(k)$ is the sequence $0,1,-1,2,-2,\ldots$.
- Then, since $\sum_{k=0}^{K} a_k e_k \to g$ as $K \to \infty$, for large K, $\sum_{k=0}^{K} a_k e_k \approx g$, and hence:

$$\begin{aligned}\int_{x\in\mathbb{S}} g(x)\,dF(x) &\approx \int_{x\in\mathbb{S}} \sum_{k=0}^{K} a_k e_k(x)\,dF(x) \\ &= \sum_{k=0}^{K} a_k \int_{x\in\mathbb{S}} e_k(x)\,dF(x)\,.\end{aligned}$$

- If we have chosen our basis functions sensibly, $\int_{x\in\mathbb{S}} e_k(x)\,dF(x)$ will be easy to evaluate analytically.
- So, once we know the as, the integral is just a simple linear combination of known quantities.

Quadrature via series expansion: Finding the as

- Recall that the way we found the as with the Fourier basis functions was by taking the integral of the product of g with each e_k.
  - If we have to evaluate these integrals numerically, we are no better off than we were to start with. (In fact, we're much worse off, as we now have K integrals, not 1.)
  - So it is essential that we can efficiently find the as.
- Newton-Cotes uses the following trick:
  - If we are truncating the sum after K terms, then we must believe $\sum_{k=0}^{K} a_k e_k(x) \approx g(x)$ for all x.
  - If we pick $K+1$ points $x_0, \dots, x_K$, then for $j \in \{0, \dots, K\}$ we have $\sum_{k=0}^{K} a_k e_k(x_j) \approx g(x_j)$.
  - This is a linear system of $K+1$ equations in $K+1$ unknowns, which may be solved via matrix inversion.
  - Both the matrix inversion and the evaluation of $\int_{x\in\mathbb{S}} e_k(x)\, dF(x)$ may be performed "off-line", so the end result is an approximation of the form: $\int_{x\in\mathbb{S}} g(x)\, dF(x) = \sum_{j=0}^{K} w_j g(x_j)$, thanks to linearity.
- Gaussian quadrature improves upon this by optimally choosing the points in order to ensure as high accuracy as possible.

Gaussian Quadrature: The trick

- Assume:
  1. That our basis is made up of polynomials, where e_k is a polynomial of degree k.
  2. That e_k has k roots in $\mathbb{S}$.
  3. That the basis is orthogonal with respect to $F(x)$, in the sense that $\int_{x\in\mathbb{S}} e_j(x)\, e_k(x)\, dF(x) = 0$ if $j \neq k$.
  4. That $K = 2n - 1$, for some $n \in \mathbb{N}^+$.

- Now, recall, $\sum_{k=0}^{K} a_k e_k(x) \approx g(x)$. The LHS is a polynomial of degree $2n-1$, by assumption.
  - Then, by the polynomial long division algorithm we have $\sum_{k=0}^{K} a_k e_k = \left(\sum_{k=0}^{n-1} q_k e_k\right) e_n + \left(\sum_{k=0}^{n-1} r_k e_k\right)$, for some $q_0, \dots, q_{n-1}$ and $r_0, \dots, r_{n-1}$.
  - Hence: $\int_{x\in\mathbb{S}} g(x)\, dF(x) \approx \left(\sum_{k=0}^{n-1} q_k \int_{x\in\mathbb{S}} e_k(x) e_n(x)\, dF(x)\right) + \left(\sum_{k=0}^{n-1} r_k \int_{x\in\mathbb{S}} e_k(x)\, dF(x)\right) = \sum_{k=0}^{n-1} r_k \int_{x\in\mathbb{S}} e_k(x)\, dF(x)$, by orthogonality.
  - Also, if $x_1, \dots, x_n$ are the roots of e_n, for all $j \in \{1, \dots, n\}$: $g(x_j) \approx \sum_{k=0}^{K} a_k e_k(x_j) = \left(\sum_{k=0}^{n-1} q_k e_k(x_j)\right) e_n(x_j) + \left(\sum_{k=0}^{n-1} r_k e_k(x_j)\right) = \sum_{k=0}^{n-1} r_k e_k(x_j)$.
  - Thus we have n equations in n unknowns ($r_0, \dots, r_{n-1}$), despite the fact our approximation is degree K.
  - Many fewer function evaluations are required for the same accuracy, as given we are integrating, we don't need to know the qs!

Gaussian Quadrature: Forms

- The set of basis polynomials satisfying the orthogonality restriction will depend on F.
  - If F is uniform, the Legendre polynomials work.
  - If F is beta-distributed, the Jacobi polynomials work.
  - If F is exponential, then the Laguerre polynomials work.
  - If F is gamma distributed, then the Generalized-Laguerre polynomials work.
  - If F is normally distributed, then the Hermite polynomials work.
  - If F has the p.d.f. $\frac{1}{\pi\sqrt{1-x^2}}$, then the Chebyshev polynomials work.

- Of course, we can always convert from one rule to another if the p.d.f.s exist.
  - For example: $\int_{x\in\mathbb{S}} g(x)\,dF_1(x) = \int_{x\in\mathbb{S}} g(x)f_1(x)\,dx = \int_{x\in\mathbb{S}} g(x)\frac{f_1(x)}{f_2(x)}f_2(x)\,dx = \int_{x\in\mathbb{S}} g(x)\frac{f_1(x)}{f_2(x)}dF_2(x)$.
  - However, if F is given by one of the distributions above, it normally delivers higher accuracy to use the corresponding rule, as the ratio of p.d.f.s is unlikely to be well approximated by a polynomial.
  - One possible reason for deviation from this maxim is that the Chebyshev polynomials come close to the minimax approximation, so in some cases using Chebyshev polynomials may provide higher accuracy.

Gauss-Hermite Quadrature

- Gaussian Quadrature with Hermite polynomials is called Gauss-Hermite Quadrature.
- Given the prevalence of Normal shocks in macro, it's by far the most common.
- The first few Hermite polynomials are:

$$e_0(x) = 1, \qquad e_1(x) = 2x,$$
$$e_2(x) = 4x^2 - 2, \qquad e_3(x) = 8x^3 - 12x,$$
$$e_4(x) = x^4 - 6x^2 + 3$$

- The approximation is usually written in the un-normalized form:

$$\int_{-\infty}^{\infty} g(x) e^{-x^2}\, dx \approx \sum_{i=1}^{n} w_i g(x_i)\,,$$

- where the weights and nodes are as given on the right (for small n).
    - Source: http://mathworld.wolfram.com/Hermite-GaussQuadrature.html

n	x	w
2	$\pm\frac{1}{2}\sqrt{2}$	$\frac{1}{2}\sqrt{\pi}$
3	0	$\frac{2}{3}\sqrt{\pi}$
	$\pm\frac{1}{2}\sqrt{6}$	$\frac{1}{6}\sqrt{\pi}$
4	$\pm\sqrt{\frac{3-\sqrt{6}}{2}}$	$\frac{\sqrt{\pi}}{4(3-\sqrt{6})}$
	$\pm\sqrt{\frac{3+\sqrt{6}}{2}}$	$\frac{\sqrt{\pi}}{4(3+\sqrt{6})}$

Cubature

- Quadrature in multiple dimensions is usually called “cubature”.
- One algorithm for cubature is to perform nested Quadrature.
- So, if we want to evaluate $\int_{x_1 \in \mathbb{S}_1} \int_{x_2 \in \mathbb{S}_2} g(x_1, x_2)\, dF_2(x_2)\, dF_1(x_1)$, for each outer node $x_{1,i}$, we evaluate $\int_{x_2 \in \mathbb{S}_2} g\left(x_{1,i}, x_2\right) dF_2(x_2)$ by quadrature.
  - If we use n nodes for x_1 and n for x_2, this gives n^2 evaluations of g.

- Macro-models often have many shocks, so a curse of dimensionality soon kicks in when using this to evaluate expectations.
- When using this to evaluate the predictive density of tomorrow’s state (in estimation), we are integrating over the distribution of each state, which again results in a curse of dimensionality in largish models.
- However, we can do better by being smart about where we “place” accuracy.
  - Rather than using many univariate polynomial approximations, we use one multivariate one.

Non-product monomial quadrature

- A monomial is a term like $x^4y^2z^3$, so multivariate polynomials are linear combinations of monomials.
- The degree of a monomial is the sum of its powers, e.g. 9 in this case.

- Monomial quadrature aims to be accurate for all monomials below a certain degree.
- The integral is again approximated by an expression of the form $\sum_{i=1}^{n} w_i g(x_i)$, but where now x is a vector.
- Accuracy can be remarkably high even with very few nodes.

- The next slide presents the most relevant approximations for the integral $\int_{x\in\mathbb{R}^d} g(x)e^{-x'x}\,dx$, taken from Stroud (1971), for your future use.
  - Throughout, subscript FS on a vector of length n means the set of all vectors formed by taking arbitrary permutations of the set of all vectors formed from the original vector by multiplying pointwise by elements of $\{f:\{1,\dots,d\}\to\{-1,1\}\}$.

Useful non-product monomial quadrature rules

Degree	Points	x	w
3	$2d$	$\left(\sqrt{\frac{d}{2}}, 0, \dots, 0\right)_{FS}$	$\frac{1}{2d}$
5	$2d^2 + 1$	$(0, \dots, 0)$	$\frac{2}{d+2}$
		$\left(\sqrt{\frac{d+2}{2}}, 0, \dots, 0\right)_{FS}$	$\frac{4-d}{2(d+2)^2}$
		$\left(\sqrt{\frac{d+2}{4}}, \sqrt{\frac{d+2}{4}}, 0, \dots, 0\right)_{FS}$	$\frac{1}{(d+2)^2}$
7	$\frac{4d^3 + 8d + 3}{3}$	$(0, \dots, 0)$	See Stroud (1971)
		$(u, 0, \dots, 0)_{FS}$	See Stroud (1971)
		$(v, 0, \dots, 0)_{FS}$	See Stroud (1971)
	$u = 0.165068012388578$ $v = 0.524647623275290$	$(u, u, 0, \dots, 0)_{FS}$	See Stroud (1971)
		$(v, v, 0, \dots, 0)_{FS}$	See Stroud (1971)
		$(u, u, u, 0, \dots, 0)_{FS}$	See Stroud (1971)

Application: The Cubature/Quadrature Kalman Filter

- In a series of papers Arasaratnam, Bhaumik, Haykin, and Swati introduce what they various call the Cubature Kalman Filter, the Quadrature Kalman Filter or the Cubature Quadrature Kalman Filter, along with Square Root variants, and smoothers.
  - Base reference is: http://125.235.3.98/dspace/bitstream/123456789/8865/1/Cubature%20Kalman%20Filters.pdf
- The idea is simple. They suppose that at the start of period t, the state has a Gaussian density.
  - Then the predicted next period mean and covariance may be found via an integral of the form $\int_{x\in\mathbb{R}^d} g(x)\phi(x)\,dx$ where ϕ is the Gaussian PDF, which they evaluate using the degree 3 cubature rule on the previous slide.
  - Given this, they can evaluate further integrals of the same form using the same cubature rule, in order to get a Gaussian approximation to the joint density of the future state and measurement.
  - A Gaussian approximation to next period's state then follows by the usual Kalman Filter conditioning trick.
- Not yet applied in macro, as far as I know, though it has been implemented in dynare (at second order) and in dynareOBC (at third order).

Application: The Unscented Kalman Filter

- The Unscented Kalman Filter has been around longer than the Cubature one, despite being more complicated.
- A point set is chosen so as to exactly recover some number of moments of the prior density of the state. (At least the mean and covariance, and up to the 5^{th} moment in the univariate case, given certain assumptions.)
- This is effectively used as the set of points for integration, much as in the CKF algorithm. (Though there are some subtleties.)
- One main difference between the CKF and the UKF is that the former (with the standard integration rule) uses $2d$ points, whereas the latter has $2d + 1$, with weight placed in the centre.
- However, in large dimensions, the weight placed on the central node is negative in the UKF, which can lead to numerical instability.

Monte Carlo Integration

- Suppose we can sample from the distribution given by $F(x)$.
  - If we can invert F, we can do this, since if U is a draw from a uniform on $[0,1]$, $F^{-1}(U)$ has the desired distribution.
- Then we can evaluate $\int_{x\in\mathbb{S}} g(x)\,dF(x)$ by drawing n samples from the distribution given by $F(x)$, $x_1,\dots,x_n$, and then using the approximation:

$$\int_{x\in\mathbb{S}} g(x)\,dF(x) \approx \frac{1}{n}\sum_{i=1}^{n} g(x_i)\,.$$

- The nice thing about Monte Carlo integration, is that no matter how many dimensions we have, by the central limit theorem, our error will be on the order of $\frac{1}{\sqrt{n}}$.
  - The error of multivariate Gauss-Hermite quadrature when $g(x)=e^x$ is on the order of $\left(\frac{e}{8n^{1/d}}\right)^{n^{1/d}}$.
  - To illustrate the difference, consider increasing n from 100 to 400.
  - This roughly halves the error under Monte Carlo integration.
  - With Gauss-Hermite the new error is roughly 3.45×10^{-982} times the old one when $d=1$.
  - But when $d=10$ it is roughly 0.54 times the old one (worse than Monte Carlo).
  - And when $d=20$, it is roughly 0.81 times the old error.

Quasi-Monte Carlo

- Although it dominates in high dimensional environments, Monte Carlo suffers in low dimensions in comparison to quadrature.
- The improvement of Monte Carlo in low dimensions may be improved by removing the randomness.

- Rather than drawing from (say) a uniform distribution, we instead take the elements of a sequence which jumps around in the unit interval in a way which ensures it covers the interval progressively better over time.
  - A so called "low discrepancy sequence".
  - One simple example of such a sequence (the van der Corput sequence) is constructed by taking each positive integer in turn, reversing its digits, then placing these digits after the decimal point. E.g. $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.01, 0.11, 0.21, 0.31, 0.41, 0.51$ ….
  - My recommended sequence is the Sobol sequence, which works well in multiple dimensions, and is built into Matlab.

- The error with QMC is on the order of $\frac{(\log n)^d}{n}$. For small d, this will be much better than $\frac{1}{\sqrt{n}}$, and its much easier to implement than Gaussian quadrature.

Markov Chain Monte Carlo: The problem

- In many situations of interest, we cannot sample directly from the distribution we're interested in.
  - For an example of the problem, suppose we have some model for which we can evaluate the likelihood i.e. $p(\mathcal{F}_T|\theta)$
    - Maybe it's a linearised DSGE model, so we can use the Kalman filter.
  - With Bayesian estimation, we are interested in evaluating

$$\mathbb{E}[g(\theta)|\mathcal{F}_T] = \int g(\theta)p(\theta|\mathcal{F}_T)\, d\theta$$

  - One way to do this would be to draw from the posterior density $p(\theta|\mathcal{F}_T)$, which is proportional to $p(\mathcal{F}_T|\theta)p(\theta)$.
  - However, even with a linearised model, $p(\mathcal{F}_T|\theta)$ will be a complicated nonlinear function of θ, so drawing from $p(\mathcal{F}_T|\theta)p(\theta)$ will be difficult.
- In the remainder of this lecture we will be looking at solutions to this problem, the first of which are Markov Chain Monte Carlo (MCMC) methods.
  - All Markov Chain Monte Carlo methods seek to find a Markov chain that we can simulate, for which the stationary distribution is the distribution of interest.
  - The most popular MCMC method is the Metropolis Hastings Algorithm, which is what we shall examine.

The Metropolis Hastings Algorithm: Set-up

- As ever, we seek to evaluate $\int_{x\in\mathbb{S}} g(x)\,dF(x)$, but we assume that F is differentiable.
- Before starting the algorithm, we choose an initial point $x_0 \in \mathbb{S}$, and a transition density $q(x'|x)$ which is a p.d.f. in $x' \in \mathbb{S}$ for all $x \in \mathbb{S}$.
  - A common choice is to take q as the density of $\mathrm{N}(x, \Sigma)$.
  - Note that in this case, $q(x'|x) = q(x|x')$ which will simplify things below.
- We also have to choose an acceptance rule $A(x'|x)$ which gives the probability of accepting a move from x' to x.
  - For the algorithm to work, we just require that for all $x, x' \in \mathbb{S}$:
$$\frac{A(x'|x)}{A(x|x')} = \frac{f(x')}{f(x)}\frac{q(x|x')}{q(x'|x)}.$$
  - A common choice satisfying this condition is:
$$A(x'|x) = \min\left\{1, \frac{f(x')}{f(x)}\frac{q(x|x')}{q(x'|x)}\right\}.$$

The Metropolis Hastings Algorithm: Algorithm (1/2)

- In step $k \in \mathbb{N}$ of the algorithm we perform the following steps:

1. Generate $\tilde{x}_{k+1}$ by drawing from $q(\tilde{x}_{k+1}|x_k)$.
2. Accept $\tilde{x}_{k+1}$ with probability $A(\tilde{x}_{k+1}|x_k)$.
   - If $\tilde{x}_{k+1}$ is accepted we set $x_{k+1} \coloneqq \tilde{x}_{k+1}$.
   - Otherwise, we set $x_{k+1} \coloneqq x_k$.

- Clearly, x_t follows a Markov-process.

The Metropolis Hastings Algorithm: Algorithm (2/2)

- A sufficient condition for x_t to have the stationary distribution π is the "detailed balance" condition that for all $x, x' \in \mathbb{S}$:
$$\pi(x)p(x'|x) = \pi(x')p(x|x').$$
- For us, $p(x'|x) = q(x'|x)A(x'|x)$, hence:
$$\frac{\pi(x)q(x'|x)}{\pi(x')q(x|x')}\frac{A(x'|x)}{A(x|x')} = 1.$$
- So, by our condition on A:
$$\frac{\pi(x)q(x'|x)}{\pi(x')q(x|x')}\frac{f(x')}{f(x)}\frac{q(x|x')}{q(x'|x)} = 1.$$
  - I.e. $\frac{\pi(x)}{f(x)} = \frac{\pi(x')}{f(x')}$. Since π and f are p.d.f.s, this can only hold if $\pi = f$.
- Thus the stationary distribution of the Markov chain we constructed is precisely the distribution of interest!

MCMC: Choice of proposal distribution

- The performance of MCMC is highly dependent on the proposal distribution, and there is little guidance on how to choose this.
- Even if we take the normal density, we still have to choose the covariance matrix.
    - $\sigma^2 I$ is a common choice, but this will perform poorly if the posterior distribution has higher variance in some directions than others, as it inevitably will.
    - Hence, a matrix of the form $\sigma^2 \Omega$ where Ω is the covariance of the prior is usually preferable.
    - $\sigma^2 \Sigma$, where Σ is the covariance of the Maximum A Posterior (MAP) estimate is another common choice.
- This still leaves σ to be chosen.
    - If σ is too low, the chain will converge very slowly, and may never explore some areas of the density.
    - If σ is too high, the acceptance rate will be very low, and so we will be performing a lot of unnecessary likelihood evaluations.
    - One common strategy is to run initial experiments with different values of σ until the desired acceptance rate is achieved.
    - The optimal acceptance rates are reported in Roberts, Gelman and Gilks (1997). When $d = 1$, the optimal acceptance rate is 50%, but for large d it decreases to around 23%.

MCMC: Further practical considerations

- Optimally, we would start our MCMC algorithm from a draw from F, however this is impossible by assumption.
- Instead, we have to wait some large number of steps for the Markov Chain to forget its initial value, and converge to its stationary distribution, discarding all draws before this point.
  - There are many convergence diagnostics for testing this, e.g. Geweke's.
  - However, convergence diagnostics can indicate false convergence if some area of the density was never explored by the Markov Chain.
  - This happens often with multi-modal densities, which are rife in DSGE models, so MCMC results should always be taken with substantial scepticism.
- Additionally, by construction, there is a high degree of persistence in the samples drawn from the chain.
  - One should always make sure that the number of samples drawn from the chain is much larger than the "forgetting time" of the chain, so the results are not affected by the chains value at the time we started taking draws.
  - Indeed, the original MH algorithm suggested only taking every K draws from the chain, where K is a measure of this forgetting time.

Importance Sampling

- Importance sampling is another solution to the problem of evaluating $\int_{x \in \mathbb{S}} g(x)\, dF(x)$ when you cannot sample from F.
- Suppose that although we cannot sample from F, we can sample from H, where both F and H are differentiable.
  - Then $\int_{x \in \mathbb{S}} g(x)\, dF(x) = \int_{x \in \mathbb{S}} g(x) \frac{f(x)}{h(x)} dH(x)$ (as we saw before).
  - So if $x_1, \dots, x_n$ are a sample drawn from H, $\int_{x \in \mathbb{S}} g(x)\, dF(x) \approx \frac{1}{n} \sum_{i=1}^{n} g(x_i) \frac{f(x_i)}{h(x_i)}$.
  - The optimal H satisfies $h(x_i) \propto g(x_i) f(x_i)$, in which case the variance of the estimator is zero. In practice, this is impossible, but it nonetheless remains the target when choosing H.
  - It is often a good idea to choose H by approximating the rate of convergence of the tails of $g(x_i) f(x_i)$.
  - It is also crucial to ensure that $h(x) > 0$ whenever $g(x) f(x) > 0$.

Sequential Monte Carlo: Set-up (1/2)

- Sequential Monte Carlo (aka the Particle Filter) is an iterative version of importance sampling, designed for nonlinear filtering problems.
  - As before x_t will be the period t state, a Markov process, and y_t will be what we observe.

- At the end of period t, suppose we have n "particles" at locations $x_{1,t}, \dots, x_{n,t}$, with importance weights $w_{1,t}, \dots, w_{n,t}$, which give an approximation to the current distribution of the state, $p(x_t|\mathcal{F}_t)$.
  - These particles may be initialized in period 0 by drawing from n draws from the prior on x_0, and setting the weights to ${}^1/_n$.

Sequential Monte Carlo: Set-up (2/2)

- Recall from the last topic that for some normalisation constant K_{t+1}:
  $p(x_{t+1}|\mathcal{F}_{t+1}) = K_{t+1} p(y_{t+1}|x_{t+1}) \int p(x_{t+1}|x_t) p(x_t|\mathcal{F}_t)\, dx_t$.
- So, using the particle approximation to $p(x_t|\mathcal{F}_t)$: $p(x_{t+1}|\mathcal{F}_{t+1}) \approx K_{t+1} p(y_{t+1}|x_{t+1}) \sum_{i=1}^{n} w_{i,t} p\left(x_{t+1}\middle|x_{i,t}\right)$.
- We wish to sample from this distribution, to form the new particle locations, but in general this will be impossible.
- Instead, suppose we have some "proposal distribution" $\pi(x_{t+1}|\mathcal{F}_{t+1}, x_t)$, from which we can sample.
  - This will play the role of H before.
  - The optimal π is given by: $\pi(x_{t+1}|\mathcal{F}_{t+1}, x_t) = p(x_{t+1}|\mathcal{F}_{t+1}, x_t) = p(x_{t+1}|y_{t+1}, x_t)$, but this will not in general be feasible.
  - A common choice is $\pi(x_{t+1}|\mathcal{F}_{t+1}, x_t) = p(x_{t+1}|x_t)$ (the "transition prior").
  - A more sophisticated approach to constructing proposal distributions is to take your favourite fast approximate non-linear Kalman filter variant (EKF, CKF, UKF etc.) and to use this as a proposal.

Sequential Monte Carlo: Basic Algorithm

- For each particle, $i \in \{1, \dots, n\}$:
  - First draw $x_{i,t+1}$ from the proposal distribution $\pi\left(x_{i,t+1} \middle| \mathcal{F}_{t+1}, x_{i,t}\right)$.
  - Then calculate the particle's candidate new weight setting:

$$\widetilde{w}_{i,t+1} = w_{i,t} \frac{p\left(y_{t+1} \middle| x_{i,t+1}\right) p\left(x_{i,t+1} \middle| x_{i,t}\right)}{\pi\left(x_{i,t+1} \middle| \mathcal{F}_{t+1}, x_t\right)}.$$

  - So when we are using the transition prior, $x_{i,t+1}$ will be a draw from $p\left(x_{i,t+1} \middle| x_{i,t}\right)$, which is easy to perform, since it is just one simulation step, and $\widetilde{w}_{i,t+1}$ will equal $w_{i,t} p\left(y_{t+1} \middle| x_{i,t+1}\right)$, meaning we do not have to be able to evaluate $p\left(x_{i,t+1} \middle| x_{i,t}\right)$.
- Then, for each particle, $i \in \{1, \dots, n\}$, set $w_{i,t+1} = \frac{\widetilde{w}_{i,t+1}}{\sum_{j=1}^{n} \widetilde{w}_{j,t+1}}$.
- If some condition is satisfied (e.g. $\sum_{j=1}^{n} w_{j,t+1}^2 > \frac{\kappa}{n}$, or just $1 = 1$!), "resample" the particles to ensure they remain concentrated in regions of high mass. I.e.:
  - For $i \in \{1, \dots, n\}$, draw $\tilde{x}_{i,t+1}$ from $\left\{x_{j,t+1} \middle| j \in \{1, \dots, n\}\right\}$, with $\Pr\left(\tilde{x}_{i,t+1} = x_{j,t+1}\right) = w_{j,t+1}$. Then replace the old particle set with $\left\{\tilde{x}_{j,t+1} \middle| j \in \{1, \dots, n\}\right\}$, and replace all the old weights with ${}^{1}/{}_{n}$.

Application: Estimating the likelihood of DSGE models via the particle filter (1/2)

- Fernandez-Villaverde and Rubio-Ramirez have been responsible for popularising this technique in macro.
  - Since with the transition prior, we only need to be able to simulate the model, this makes it very easy to implement for DSGE models.

- Additionally, in a seminal paper Andrieu, Doucet, and Holenstein (2007) proved both that the particle filter gave an unbiased estimate of the likelihood, and that unbiasedness is sufficient for correct Bayesian inferrence, when the posterior is evaluated via the Metropolis-Hastings algorithm.
  - Nonetheless, performance may be less than ideal.

Application: Estimating the likelihood of DSGE models via the particle filter (2/2)

- Although when calculating an integral via Monte Carlo the error is on the order of $\frac{1}{\sqrt{n}}$, in the particle filter, the particles are playing two distinct roles, which means the error scaling may be much worse.
  - Of course they are the set of points we use to calculate the integral $\int p(x_{t+1}|x_t)p(x_t|\mathcal{F}_t)\,dx_t$, but they are also the points we use to approximate the distribution of $p(x_t|\mathcal{F}_t)$.
  - It is easy to see that to match k moments of a d dimensional distribution we need on the order of d^k particles. So to do better than e.g. the CKF with its $2d$ nodes, we will need at least on the order of d^3 particles. For a standard medium scale model this could easily mean $10000+$ particles.
  - Furthermore, many of these particles will be "lost" each period, as they could not possibly explain the observed y_{t+1}, so the required number may be orders of magnitude higher.

- Kollman (2013) found that his approximate algorithm (for second order pruned systems) outperformed the particle filter for computationally feasible numbers of particles.

Application: Using SMC to evaluate the posterior density of a DSGE model

- Recall that with Bayesian estimation, we are interested in drawing from the posterior density, which is proportional to $p(y_T, \dots, y_1|\theta)p(\theta)$, where, as in our discussion of MCMC, we suppose that we can evaluate $p(y_T, \dots, y_1|\theta)$ somehow.
- Herbst and Schorfheide (2012) propose using SMC to evaluate from the posterior density.
  - Note, this is not using SMC to evaluate the likelihood!
- They do this by constructing a series of approximations to the posterior.
  - The approximation in step K is some weighted combination of the prior and posterior densities, where the weight on the prior gradually decreases from 100% to 0%.
  - The technical details are a little complicated, but one may think of it as repeated importance sampling where the distribution from the previous step is used as the importance weight for the next step.
- Since they start with mass evenly distributed over the prior, their method readily handles multi-modal distributions. (As more weight is placed on the posterior, humps gradually emerge, and particles cluster in these humps.)
  - As a result, they show that its performance comfortably exceeds that of MCMC.
- Their algorithm is also readily parallelizable, unlike MCMC.
  - However, it appears large numbers of likelihood evaluations are required, so it may be prohibitively expensive for large models.