15.472: Computational Methods II

Dan Greenwald

Fall 2021

Dynamic Programming

- Let x_t be endogenous states, z_t be exogenous states, and y_t be controls. The basic problem is:

$$\begin{aligned} V(x_t, z_t) &= \max_{y_t \in \Gamma(x_t, z_t)} F(x_t, y_t, z_t) + \beta E_t \left[V(x_{t+1}, z_{t+1}) \right] \\ x_{t+1} &= g(x_t, y_t, z_t) \\ z_{t+1} &= h(z_t, \varepsilon_{t+1}) \end{aligned}$$

- Example: consumption-savings problem.

$$\begin{aligned} V(a_t, w_t) &= \max_{x_t \geq 0} u\left(a_t + w_t \bar{L} - s_t\right) + \beta E_t \left[V(a_{t+1}, w_{t+1}) \right] \\ a_{t+1} &= R s_t \\ \log w_{t+1} &= (1-\rho) \log \bar{w} + \rho \log w_t + \varepsilon_{t+1}, \qquad \varepsilon_{t+1} \sim N(0, \sigma^2) \end{aligned}$$

Discrete Dynamic Programming

- Very simple and robust approach: assume $x_t \in \mathcal{X} = (\bar{x}_1, \ldots, \bar{x}_N)$, $z_t \in \mathcal{Z} = (\bar{z}_1, \ldots, \bar{z}_K)$.
  - Easy to estimate time series using Hamilton filter (see Farmer, 2017).

- Basic problem reframed:

$$V(x_t, z_t) = \max_{x_{t+1} \in \Gamma(x_t, z_t)} F(x_t, z_t, x_{t+1}) + \beta \sum_{z_{t+1}} P(z_{t+1}|z_t) V(x_{t+1}, z_{t+1})$$

- Effects of discretization:
  - Choose x_{t+1} directly instead of y_t (can't leave grid).
  - Expectation is matrix multiplication.

- Notation: $\mathbf{X}(a, b; c)$ is a matrix where the columns stack over a and b (i.e., $(a_1, b_1), (a_1, b_2), \ldots, (a_2, b_1), \ldots$) and the rows stack over c.

Discrete Dynamic Programming

- **Step 1:** given iteration k guess $\mathbf{V}_k$, optimize decision.
- Define

$$\mathbf{Q}(x_t, z_t; x_{t+1}) = \underbrace{\mathbf{F}(x_t, z_t; x_{t+1})}_{NK \times N} + \beta \Big(\underbrace{\mathbf{P}(z_t; z_{t+1})}_{K \times K} \otimes \underbrace{\mathbf{1}_N}_{N \times 1} \Big) \underbrace{\mathbf{V}(x_{t+1}; z_{t+1})'}_{K \times N}$$

  if $x_{t+1} \in \Gamma(x_t, z_t)$, and $-\infty$ otherwise.
- Reminder: $\otimes$ is the Kronecker product, so that

$$A \otimes B = \begin{bmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{n1}B & \cdots & A_{nn}B \end{bmatrix}$$

- Define $x^*_{t+1}(x_t, z_t) = \arg\max_{x_{t+1}} \mathbf{Q}(x_t, z_t; x_{t+1})$. This is the column-wise max.

Discrete Dynamic Programming

- **Step 2:** given decisions, update $\mathbf{V}$ ("Howard Improvement").
  - Can update $\mathbf{V}_{k+1}$ by plugging in x^*_{t+1} and $\mathbf{V}_k$ on the RHS, then iterate, but this is slow.
  - Better approach: solve for **exact** value function under policy x^*_{t+1}.

- Define:

$$\begin{aligned} A(x_t, z_t, x_{t+1}, z_{t+1}) &= P(z_{t+1}|z_t) \cdot \mathbf{1}\left\{x_{t+1} = x^*_{t+1}(x_t, z_t)\right\} \\ F^*(x_t, z_t) &= F(x_t, z_t, x^*_{t+1}) \end{aligned}$$

- Then we have:

$$\underbrace{\mathbf{V}(x_t, z_t)}_{NK\times 1} = \underbrace{\mathbf{F}^*(x_t, z_t)}_{NK\times 1} + \beta \underbrace{\mathbf{A}(x_t, z_t; x_{t+1}, z_{t+1})}_{NK\times NK} \underbrace{\mathbf{V}(x_{t+1}, z_{t+1})}_{NK\times 1}$$

  which implies the exact solution $\mathbf{V}_{k+1} = (I - \beta\mathbf{A})^{-1}\mathbf{F}^*$.

Discrete Dynamic Programming

- Iterate on Steps 1 and 2 until x^*_{t+1} stops changing. Then you are done!
- Stationary distribution: eigenvector of $\mathbf{A}'$ associated with unit eigenvalue.
  - Similarly, stationary dist. of exogenous states is eigenvector of P' with unit eigenvalue.
- Note: $\mathbf{A}$ will contain many zeros, often better to use sparse matrices.
- For P, use Rouwenhorst (1995) method to approximate Gaussian AR(1) processes.
  - Other approximations struggle as $\rho \to 1$.
  - Better to read treatment in Kopecky and Suen, RED 2010.
- Suffers from curse of dimensionality, but GPUs can provide huge speedup!

Special Case: Exogenous Asset Pricing

- Assume that all states are exogenous.
- Combine $E_t[M_{t+1}R_{t+1}] = 1$ and definition $R_{t+1} = (P_{t+1} + D_{t+1})/P_t$ to obtain

$$PD(z_t) = E_t \left\{ M(z_t, z_{t+1}) \left(PD(z_{t+1}) + 1 \right) \frac{D(z_{t+1})}{D(z_t)} \right\}.$$

- Then we can solve for PD exactly with a **single linear inversion**:

$$\begin{aligned} A(z_t, z_{t+1}) &\equiv P(z_t, z_{t+1}) M(z_t, z_{t+1}) \frac{D(z_{t+1})}{D(z_t)} \\ \mathbf{PD}(z_t) &= \mathbf{A}(z_t; z_{t+1}) \Big(\mathbf{PD}(z_{t+1}) + \mathbf{1}_K \Big) \\ \mathbf{PD} &= (I - \mathbf{A})^{-1} \mathbf{1}_K \end{aligned}$$

Generic Optimality Conditions

- As long as the problem is well-behaved (uniquely determined by FOCs), it is usually better to solve the FOCs than to directly use the value function.
- Typical approach is to just start taking derivatives, but can actually be more systematic.
- Let's add some additional structure (slight change of notation):
  - Let c_t be consumption, and y_t be all **other** controls.
  - Let $\Psi(x_t, c_t, y_t, z_t) \geq 0$ be the budget constraint, and $\Gamma(x_t, c_t, y_t, z_t) \geq 0$ be all **other** constraints.
  - Assume the budget constraint is written $c_t \leq \cdots$ so that $\partial \Psi_t / \partial c_t = -1$.
  - Let $F(x_t, y_t, z_t) = u(x_t, c_t, y_t, z_t)$.

Generic Optimality Conditions

- Generic optimality condition for y_t:

$$0 = \underbrace{\left(\frac{\partial u_t}{\partial c_t}\right)^{-1} \frac{\partial u_t}{\partial y_t}}_{\text{utility}} + \underbrace{\frac{\partial \Psi_t}{\partial y_t}}_{\text{resources}} + \underbrace{\mu_t \frac{\partial \Gamma_t}{\partial y_t}}_{\text{constraints}} + \underbrace{\Omega_t \frac{\partial x_{t+1}}{\partial y_t}}_{\text{continuation}}$$

- All quantities expressed in units of consumption.
- Marginal continuation values Ω_t defined by fixed point

$$\Omega_t = E_t \left\{ M_{t+1} \left[\left(\frac{\partial u_{t+1}}{\partial c_{t+1}}\right)^{-1} \frac{\partial u_{t+1}}{\partial x_{t+1}} + \frac{\partial \Psi_{t+1}}{\partial x_{t+1}} + \mu_{t+1} \frac{\partial \Gamma_{t+1}}{\partial y_{t+1}} + \Omega_{t+1} \frac{\partial x_{t+2}}{\partial x_{t+1}} \right] \right\}$$

where M_{t+1} is the SDF. **Note: works for EZW preferences.**

Example: Kaltenbrunner and Lochstoer (2010)

- Preferences: $U_t = \left((1-\beta) C_t^{1-\rho} + \beta E_t \left[U_{t+1}^{1-\gamma} \right]^{\frac{1-\rho}{1-\gamma}} \right)^{\frac{1}{1-\rho}}$
- Budget constraint: $C_t \leq Z_t^{1-\alpha} K_t^{\alpha} - i_t K_t.$
- Endogenous state LOM: $K_{t+1} = (1-\delta) K_t + \phi(i_t) K_t.$
- Exogenous state LOM: $\log Z_{t+1} = \phi \log Z_t + \varepsilon_{t+1}.$
- Optimality conditions:

$$0 = -1 + \phi'(i_t)\Omega_{K,t}$$

$$\Omega_{K,t} = E_t \left\{ M_{t+1} \left[\alpha \left(\frac{Z_{t+1}}{K_{t+1}} \right)^{1-\alpha} - i_{t+1} + \Big((1-\delta) + \phi(i_{t+1}) \Big) \Omega_{K,t+1} \right] \right\}$$

Example: Kaltenbrunner and Lochstoer (2010)

- Preferences: $U_t = \left((1-\beta) C_t^{1-\rho} + \beta E_t \left[U_{t+1}^{1-\gamma} \right]^{\frac{1-\rho}{1-\gamma}} \right)^{\frac{1}{1-\rho}}$
- Budget constraint: $C_t \leq Z_t^{1-\alpha} K_t^{\alpha} - i_t K_t.$
- Endogenous state LOM: $K_{t+1} = (1-\delta) K_t + \phi(i_t) K_t.$
- Exogenous state LOM: $\log Z_{t+1} = \phi \log Z_t + \varepsilon_{t+1}.$
- Optimality conditions:

$$
\begin{aligned}
1 &= E_t \left[M_{t+1} R_{t+1} \right] \\
R_{t+1} &\equiv \frac{\alpha (Z_{t+1}/K_{t+1})^{1-\alpha} - i_{t+1} + \Big((1-\delta) + \phi(i_{t+1}) \Big) q_{t+1}}{q_t} \\
q_t &\equiv \phi'(i_t)^{-1}
\end{aligned}
$$

Complementary Slackness

- Complementary slackness: given constraint Γ_t and multiplier μ_t:

$$\mu_t \Gamma_t = 0, \qquad \mu_t \geq 0, \qquad \Gamma_t \geq 0.$$

- Example: lower bound $y_t \geq 0$.
  - Challenge: kinked, nondifferentiable policy function.
- **Auxiliary variable** (Garcia and Zangwill) approach:
  - Define policy function as auxiliary variable α_t.
  - Define $y_t = \max(\alpha_t, 0)^k$ for $k > 1$.
  - Define $\mu_t = \max(-\alpha_t, 0)^k$ for $k > 1$.
- Delivers continuously differentiable policy function.

Time Iteration

- Assume equilibrium conditions follow $f\left(x, y, z, \mathcal{E}(x, y, z)\right) = 0$, where

$$\mathcal{E}_t = E_t\left[q(x_{t+1}, y_{t+1}, z_{t+1})\right]$$

- Choose grid $\{\bar{x}_i, \bar{z}_i\}$ and basis functions $\psi(s, z)$.
- Let b_k be the coefficients from the previous (kth) iteration.
- Key idea: use previous guess b_k to form expectations $\mathcal{E}$:

$$\begin{aligned} y_{t+1} &= b_k' \psi(x_{t+1}, z_{t+1}) \\ \mathcal{E}_t &= \sum_j \omega_j q(x_{t+1}, y_{t+1}, z_{t+1}) \end{aligned}$$

where ω_j are quadrature weights, then solve for y_t given $\mathcal{E}_t$.

Time Iteration on Controls

- **Time iteration on controls**: for each $(\bar{x}_i, \bar{z}_i)$, choose y_i so that $f\left(\bar{x}_i, y_i, \bar{z}_i, \mathcal{E}(\bar{x}_i, \bar{z}_i; b_k)\right) = 0$, with

$$\mathcal{E}_t(\bar{x}_i, \bar{z}_i; b_k) = \sum_j \omega_j q\Big(x_{t+1}, b_k' \psi(x_{t+1}, z_{t+1}), z_{t+1}\Big)$$

- Recipe: given y_i, quadrature node $\bar{\varepsilon}_j$, compute

$$\begin{aligned} x_{t+1} &= g(\bar{x}_i, y_i, \bar{z}_i) \\ z_{t+1} &= h(\bar{z}_i, \bar{\varepsilon}_j). \end{aligned}$$

- Use nonlinear equation solver. Gradient:

$$\frac{df}{dy_t} = \frac{\partial f}{\partial y_t} + \frac{\partial f}{\partial \mathcal{E}_t} E_t\left[\frac{\partial x_{t+1}}{\partial y_t}\left(\frac{\partial q}{\partial x_{t+1}} + \frac{\partial q}{\partial y_{t+1}}\frac{\partial y_{t+1}}{\partial x_{t+1}}\right)\right]$$

  where $\partial y_{t+1}/\partial x_{t+1} = b_k'(\partial \psi(x_{t+1}, z_{t+1})/\partial x_{t+1})$.

- Once solutions $\{y_i^*\}$ have been found, choose b_{k+1} so that $y_i^* = b_{k+1}'\psi(\bar{x}_i, \bar{z}_i)\ \forall i$, repeat.

Time Iteration

- Iterating on controls is straightforward but not very efficient.
  - Additional challenge: y_{t+1} may not be non-smooth in states.
- Alternative: **time iteration on coefficients**. Solve directly for $\hat{b}$ that satisfies

$$f\left(\bar{x}_i, \hat{b}'\psi(\bar{x}_i, \bar{z}_i), \bar{z}_i, \mathcal{E}\right) = 0.$$

  i.e., compute $y_i = \hat{b}'\psi(\bar{x}_i, \bar{z}_i)$ and proceed as before.
- Use nonlinear equation solver with gradient

$$\frac{\partial f}{\partial \hat{b}} = \left\{ \frac{\partial f}{\partial y_t} + \frac{\partial f}{\partial \mathcal{E}_t} E_t \left[\frac{\partial x_{t+1}}{\partial y_t} \left(\frac{\partial q}{\partial x_{t+1}} + \frac{\partial q}{\partial y_{t+1}} \frac{\partial y_{t+1}}{\partial x_{t+1}} \right) \right] \right\} \frac{\partial y_t}{\partial \hat{b}}$$

  where $\partial y_t / \partial \hat{b} = \psi(x_t, z_t)'$.
- Update $b_{k+1} = \hat{b}$ and repeat until $||b_{k+1} - b_k||$ is smaller than some threshold.

Direct Solution

- Most efficient (but least robust): solve for b directly.
- Apply same solution to both sides:

$$0 = f\left(x_t, b'\psi(x_t, z_t), z_t, \mathcal{E}_t\right)$$
$$\mathcal{E}_t = E_t\left[q\left(x_{t+1}, b'\psi(x_{t+1}, z_{t+1}), z_{t+1}\right)\right].$$

- Run nonlinear equation solver with gradient

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial y_t}\frac{\partial y_t}{\partial b} + \frac{\partial f}{\partial \mathcal{E}_t}E_t\left[\left(\frac{\partial q}{\partial x_{t+1}} + \frac{\partial q}{\partial y_{t+1}}\frac{\partial y_{t+1}}{\partial x_{t+1}}\right)\frac{\partial x_{t+1}}{\partial y_t}\frac{\partial y_t}{\partial b} + \frac{\partial q}{\partial y_{t+1}}\frac{\partial y_{t+1}}{\partial b}\right]$$

- Can start with time iteration and then switch to direct solution to get best of both worlds.

Nonlinear Optimizers

- Solution techniques just discussed require nonlinear equation solver (or optimizer).
- Newton's method: for general system of nonlinear equations with solution $F(x^*) = 0$, use Taylor expansion $0 = F(x^*) \simeq F(x) + \nabla F(x)(x^* - x)$ to obtain

$$x^* \simeq x + \Delta, \qquad \Delta \equiv \underbrace{-\left(\nabla F(x)\right)^{-1} F(x^*)}_{\text{"Newton step"}}$$

- Use backstepping to avoid getting stuck:
  - Start with step of size $s = \Delta$, then keep cutting s in half until $||f(x+s)|| < ||f(x)||$.
- Sign of good convergence: quadratic approach (e.g., `..., ?e-2, ?e-4, ?e-8`).
- Very good open source solver: IPOPT.
  - Can code equation solver as minimizing $g(x) = 1$ s.t. constraints $f(x) = 0$.

Special Case: Exogenous Asset Pricing

- Return to special case

$$PD(z_t) = E_t \left\{ M(z_t, z_{t+1}) \left(PD(z_{t+1}) + 1 \right) \frac{D(z_{t+1})}{D(z_t)} \right\}.$$

- Apply guess $PD(z_t) = \psi(z_t)'b$, and use quadrature scheme $(\omega_j, \bar{\varepsilon}_j)$:

$$\psi(z_t)'b = \sum_j \omega_j M(z_t, \bar{\varepsilon}_j) \left(\frac{D(z_t, \bar{\varepsilon}_j)}{D(z_t)} \right) \left(\psi(z_t, \bar{\varepsilon}_j)'b + 1 \right)$$

with slight abuse of notation to substitute $(z_t, \bar{\varepsilon}_j)$ for z_{t+1}.

Special Case: Exogenous Asset Pricing

- If we now define

$$A(z_t) = \sum_j \omega_j M(z_t, \bar{\varepsilon}_j) \left(\frac{D(z_t, \bar{\varepsilon}_j)}{D(z_t)} \right) \psi(z_t, \bar{\varepsilon}_j)$$

$$c(z_t) = \sum_j \omega_j M(z_t, \bar{\varepsilon}_j) \left(\frac{D(z_t, \bar{\varepsilon}_j)}{D(z_t)} \right)$$

  then we obtain

$$\Psi b = \mathbf{A} b + \mathbf{c}$$

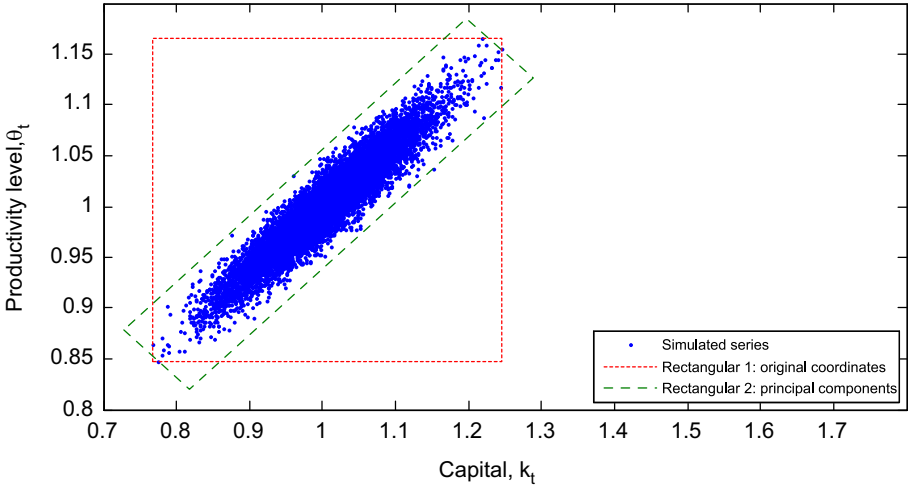
$$b = (\Psi - \mathbf{A})^{-1} \mathbf{c}$$

- Another one-step linear solution!

Additional Refinements

- **Precomputation**:
  - Can save time by pre-computing $\psi(\bar{x}_i, \bar{z}_i)$.
  - Note that for a given grid $\{\bar{z}_i\}$ and nodes $\{\bar{\varepsilon}_j\}$, end up with the same grid over z_{t+1}.
  - If $\psi(x_{t+1}, z_{t+1}) = \psi_x(x_{t+1})\psi_z(z_{t+1})$, then we can also precompute $\psi_z(z_{t+1}(\bar{z}_i, \bar{\varepsilon}_j))$.
- **Adaptive domain**:
  - Approximations work better when variables are not highly correlated.
  - Better: use principal components as states.
  - Do SVD to obtain $X = (x_1, \ldots, x_T)' = USV'$, then the PCs are $\tilde{X} = XV$.
  - Recover X using $X = \tilde{X}V'$.

Adaptive Domain: Illustration



Source: Judd, Maliar, Maliar, Valero (2013).

Perturbation Methods

- Based on **local** expansion around a point.
- Computationally cheap, but less accurate far from approximation point.
  - Great as initial guess for global solution.
- Fold exogenous states into x_t to rewrite

$$\begin{aligned} 0 &= E_t\left[f(x_t, y_t, x_{t+1}, y_{t+1})\right] \\ y_t &= g(x_t, \sigma) \\ x_{t+1} &= h(x_t, \sigma) + \sigma\eta\varepsilon_{t+1}. \end{aligned}$$

- Note that g and h are different from earlier notation.

Perturbation Methods

- First order perturbation:

$$y_t = g(\bar{x}, 0) + g_x(\bar{x}, 0)(x - \bar{x}) + g_\sigma(\bar{x}, 0)\sigma$$
$$x_{t+1} = h(\bar{x}, 0) + h_x(\bar{x}, 0)(x - \bar{x}) + h_\sigma(\bar{x}, 0)\sigma + \sigma\eta\varepsilon_{t+1}$$

- Choose values to set derivatives of equilibrium condition to zero:

$$(x_t): \qquad 0 = \mathbf{f_x} + \mathbf{f_y}\mathbf{g_x} + \mathbf{f_{x'}}\mathbf{h_x} + \mathbf{f_{y'}}\mathbf{g_x}\mathbf{h_x} \tag{1}$$
$$(\sigma): \qquad 0 = \mathbf{f_y}\mathbf{g}_\sigma + \mathbf{f_{x'}}\mathbf{h}_\sigma + \mathbf{f_{y'}}\left(\mathbf{g}_\sigma + \mathbf{g_x}\mathbf{h}_\sigma\right) \tag{2}$$

- Solution to (2) implies $g_\sigma = h_\sigma = 0$ (no risk effects).

- Apply $g(\bar{x}, 0) = \bar{y}, h(\bar{x}, 0) = \bar{x}$, define e.g., $\hat{x} = x - \bar{x}$, to obtain system that must solve (1):

$$\hat{y}_t = g_x\hat{x}_t$$
$$\hat{x}_{t+1} = h_x\hat{x}_t + \sigma\eta\varepsilon_{t+1}$$

- Many solution techniques: Sims (2001), Klein (2000).

Perturbation Methods

- Second-order perturbation (see e.g., Judd and Guu (1997) for solution method):

$$\hat{y}_t = \mathbf{g}_{\mathbf{x}}\hat{x}_t + \frac{1}{2}\mathbf{G}_{xx}(\hat{x}_t \otimes \hat{x}_t) + \frac{1}{2}\mathbf{g}_{\sigma\sigma}\sigma^2$$
$$\hat{x}_{t+1} = \mathbf{h}_{\mathbf{x}}\hat{x}_t + \frac{1}{2}\mathbf{H}_{xx}(\hat{x}_t \otimes \hat{x}_t) + \frac{1}{2}\mathbf{h}_{\sigma\sigma}\sigma^2 + \sigma\eta\varepsilon_{t+1}$$

- Now risk influences policy functions (in a constant way).
  - Third-order: $\sigma^2\hat{x}_t$ term linear in states.
  - Higher order: nonlinear risk-state interactions.

- Major problem: explosiveness. Univariate example:

$$\hat{x}_{t+1} = \cdots + \frac{1}{2}h_{xx}\hat{x}_t^2 = \cdots + \frac{1}{2}h_{xx}\left(\cdots + \frac{1}{2}h_{xx}\hat{x}_{t-1}^2\right)^2$$

Pruned State Space

- **Pruned state-space** approach (Andreasen et al, 2018). Split x_t into first-order terms x_t^f and second-order terms x_t^s:

$$\hat{x}_{t+1}^f = \mathbf{h_x}\hat{x}_t^f + \sigma\eta\varepsilon_{t+1}$$
$$\hat{x}_{t+1}^s = \mathbf{h_x}\hat{x}_t^s + \frac{1}{2}\mathbf{H}_{xx}(\hat{x}_t^f \otimes \hat{x}_t^f) + \frac{1}{2}\mathbf{h}_{\sigma\sigma}\sigma^2$$

- No interaction between x^s and x^s means no explosiveness: $\hat{x}_t = A(L)\varepsilon_t + B(L)\varepsilon_t^2$.

- Policy functions:

$$\hat{y}_t = \mathbf{g_x}\left(\hat{x}_t^f + \hat{x}_t^s\right) + \frac{1}{2}\mathbf{G}_{xx}(\hat{x}_t^f \otimes \hat{x}_t^f) + \frac{1}{2}\mathbf{g}_{\sigma\sigma}\sigma^2$$

- See paper for third-order equivalent.

Computational Advice

- Test as exhaustively as possible.
  - Compare to case with closed form/known/approximate solution.
  - Compare to previous version if you are making changes.
  - Step through new code to make sure it is doing what you think.
- Fixing bugs has priority over everything else.
  - If something weird happens but then "goes away on its own," go after it!
- Write code generically and store bug-free functions in libraries. **Avoid copy-paste!**
- Use version control (e.g., git) to back up files.
  - If bug emerges, find last working snapshot to isolate error.