Advanced Topics in System Identification

7.1: Closed-loop system identification

- We've now covered the basic major areas of LTI system identification: Unit-pulse-response models, frequency-response models, transfer-function models, and state-space models.
- For this last chapter we consider some more advanced topics that are actually fairly simple alterations to what we have already studied.
- These include: Identifying systems configured in a feedback loop; reducing model order; performing real-time estimation using ARX models; and handling static nonlinearities.

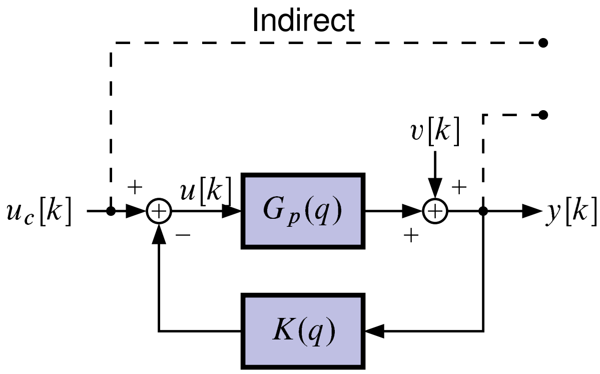
Closed-loop system identification

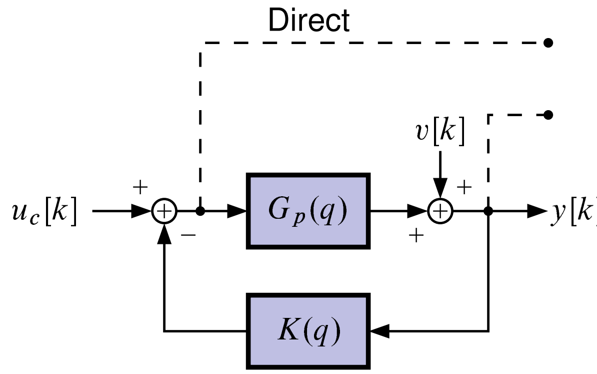
- The system to be identified may be operating in closed loop, so only closed-loop data are available.
- Or, the system may be unstable, so it is necessary to stabilize it via feedback to be able to get finite measurements.
- Key point is that the input $u[k]$ is now related to $y[k]$ through the controller. There are two relationships:

$$y[k] = G(q)u[k] + v[k], \quad \text{and} \quad u[k] = -K(q)y[k] + u_c[k].$$

- Input and sensor noise now correlated.
- Cannot apply standard tools directly. Must develop slightly different methods to account for noise correlations correctly.

- Will briefly consider two transfer-function approaches: Indirect, direct.





- The signal $u_c[k]$ is a reference input that we can use to excite the closed-loop system: $u[k] = -K(q)y[k] + u_c[k]$.

INDIRECT: In the indirect approach, we

- Apply the signal $u_c[k]$; measure $y[k]$ and $u_c[k]$.
- Develop a model for the closed-loop system, assuming $K(q)$ LTI

$$y[k] = G_{cl}(q)u_c[k] = (I + G_p(q)K(q))^{-1}G_p(z)u_c[k].$$

- Assuming $K(q)$ is known, $G_p(q) = G_{cl}(q)(I - K(q)G_{cl}(q))^{-1}$.
- Similarly, in terms of the disturbance,

$$y[k] = H_{cl}(q)e[k] = (I + G_p(q)K(q))^{-1}H(q)e[k],$$

so we can find that $H(q) = (I + G_p(q)K(q))H_{cl}(q)$.

DIRECT: In the direct approach, we

- Apply the signal $u_c[k]$; measure $y[k]$ and $u[k]$.
- Identify $G_p(z)$ directly from $y[k]$ and $u[k]$.

- Concerns:

DIRECT: How well can we shape $u[k]$? Can we make it persistently exciting? Can we avoid frequencies that are nulled out?

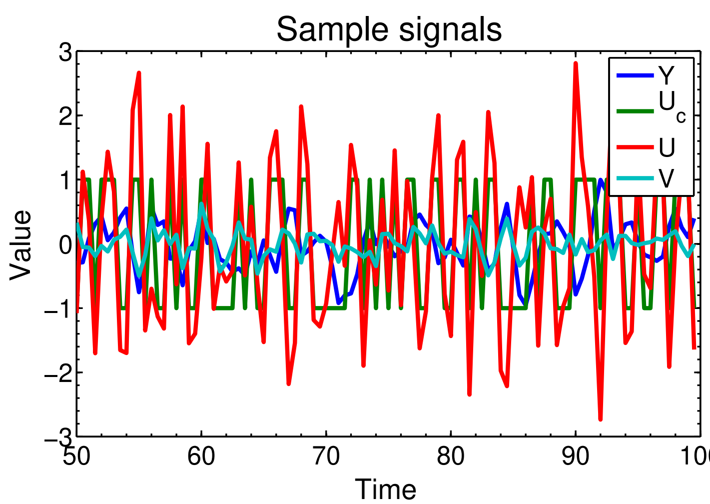
INDIRECT: Does a good model of $G_{cl}(q)$ translate into a good model of $G_p(q)$? Is the controller really linear?

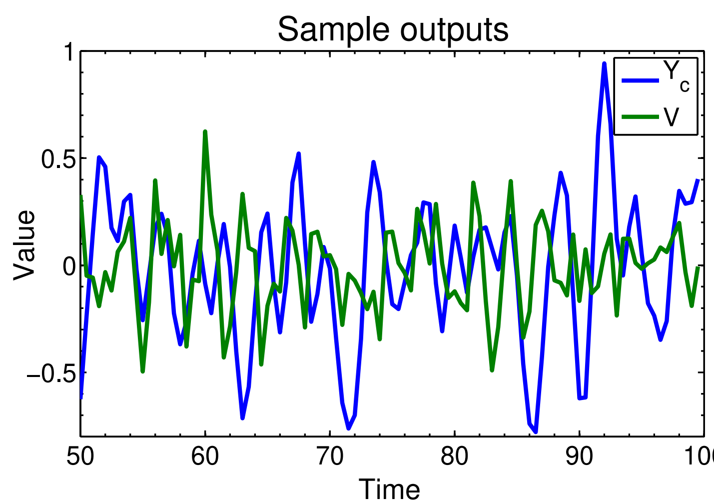
- I suggest that you try both and compare... In the next sections, we use an example to look at the two approaches.

EXAMPLE: Open-loop system has $y[k] = G_{ol}(q)u[k] + H_{ol}e[k]$.

$$G_{ol}(q) = \frac{0.11q^{-1} + 0.10q^{-2}}{1 - 1.5q^{-1} + 0.78q^{-2}}, \quad \text{and} \quad H_{ol}(q) = \frac{1 - 1.63q^{-1} + 0.76q^{-2}}{1 - 1.86q^{-1} + 0.99q^{-2}}.$$

- Design a simple controller, $K(q) = 2.2\dfrac{1 - 0.75q^{-1}}{1 - 0.25q^{-1}}$.
- Since all numerators/denominators are written with equal length, we can switch back and forth from transfer-function to polynomial format in MATLAB. (But, be careful with this!)
- The input signals to the system are plotted below, where
  - $y[k]$ is the (disturbed, measured) plant output;
  - $u_c[k]$ is the user-generated control input;
  - $u[k]$ is the input to the plant, including the feedback;
  - $v[k]$ is the disturbance added to the plant output; and
  - $y_c[k]$ is the "clean" nominal plant output without disturbance.

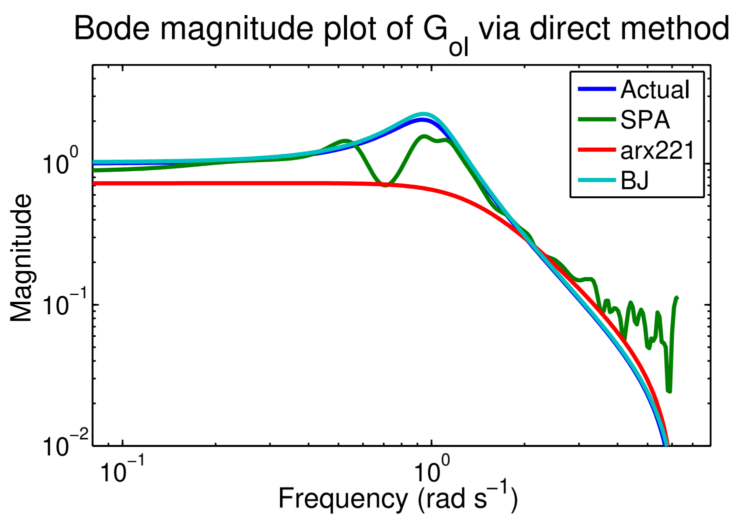


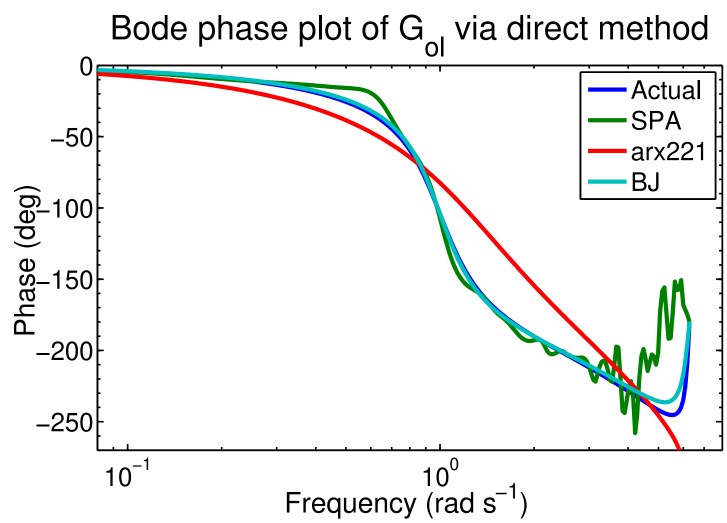


7.2: Results of closed-loop system ID example

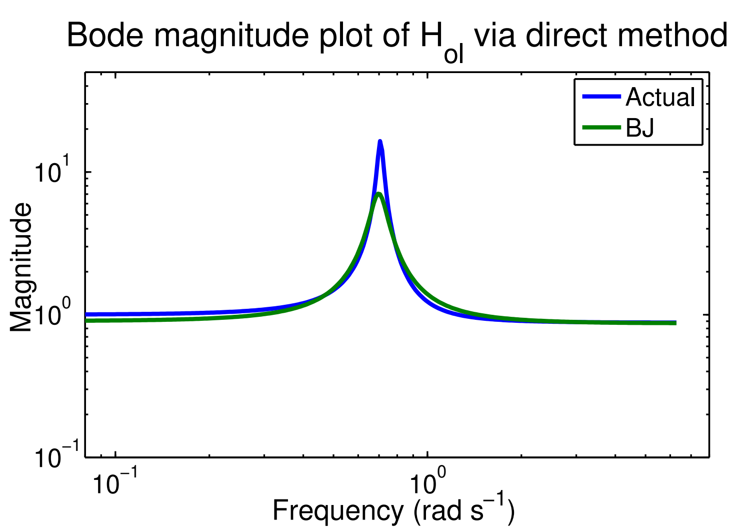
Direct method

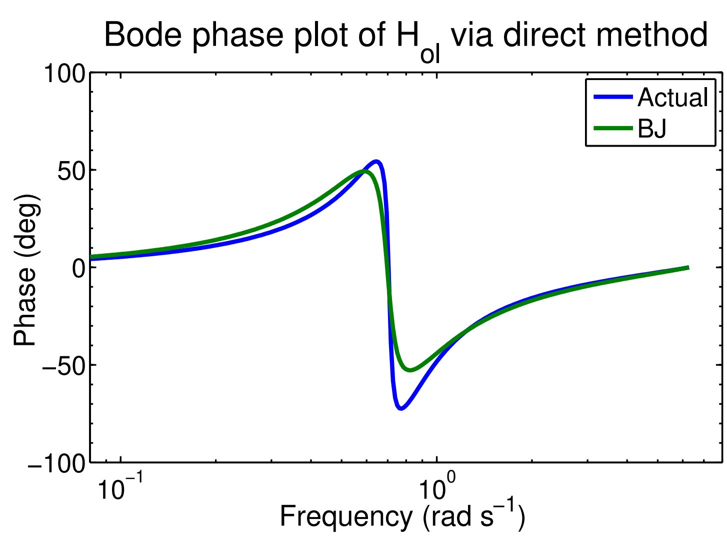
- The direct method was used to identify $G(q)$ using SPA, ARX, and BJ.
  - Notice that SPA is quite biased by the disturbance, which has a lot of energy around 0.7 rads/sec.



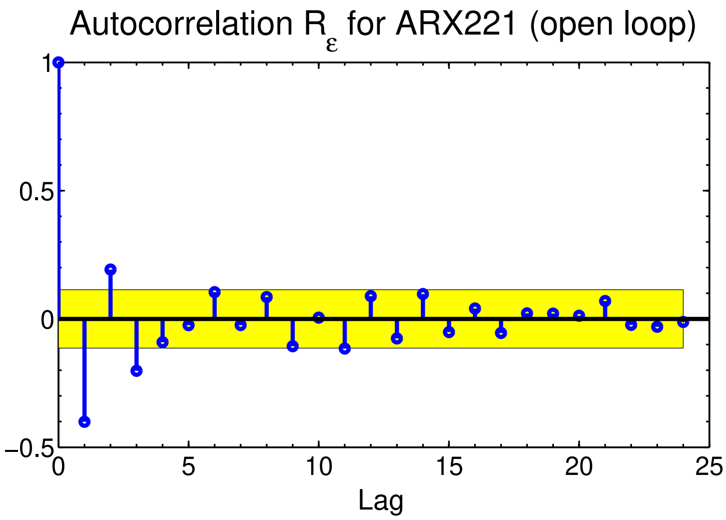


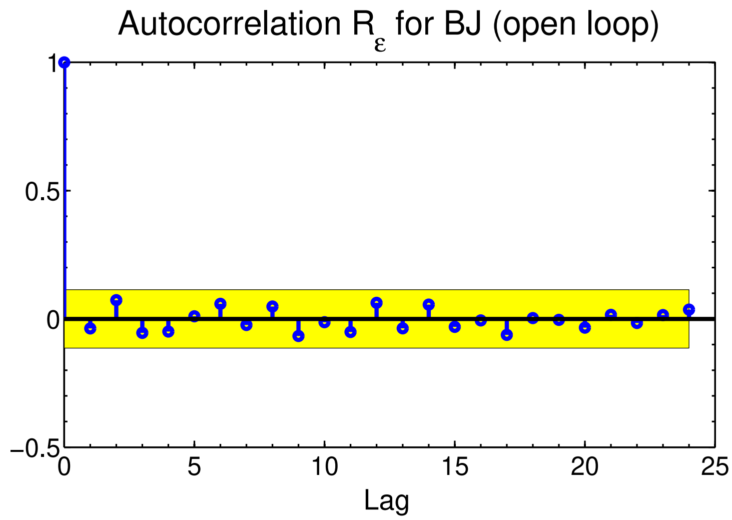
- Of the above, only BJ produces an estimate of $H(q)$.

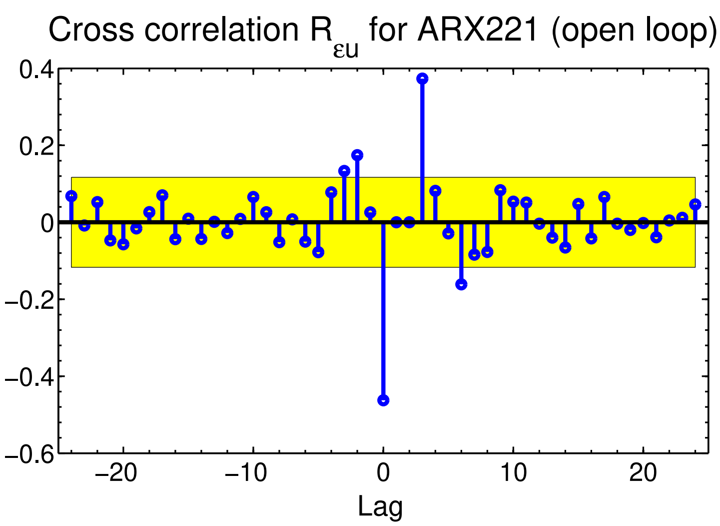


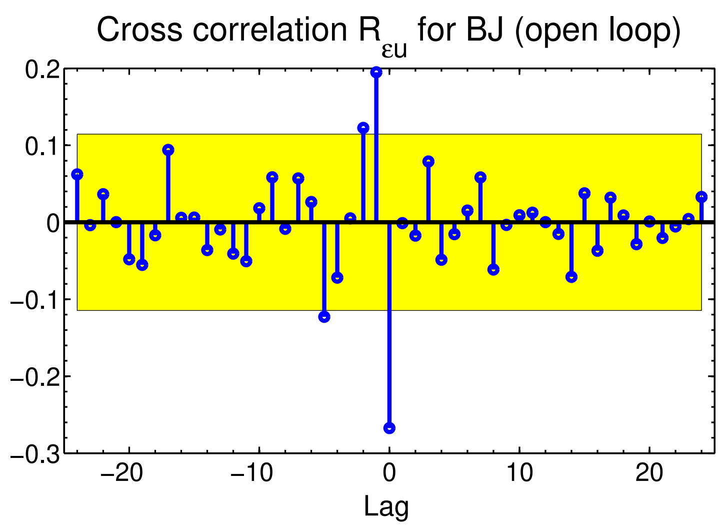


- Residual analysis shows that BJ is better than ARX, but still not great for small lags.

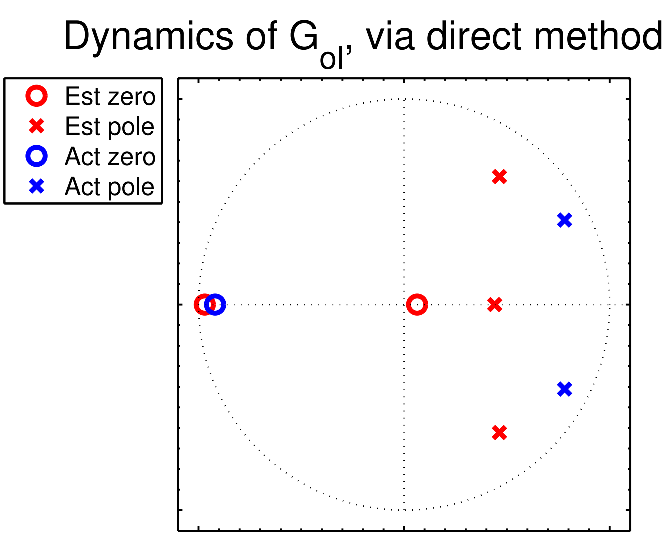


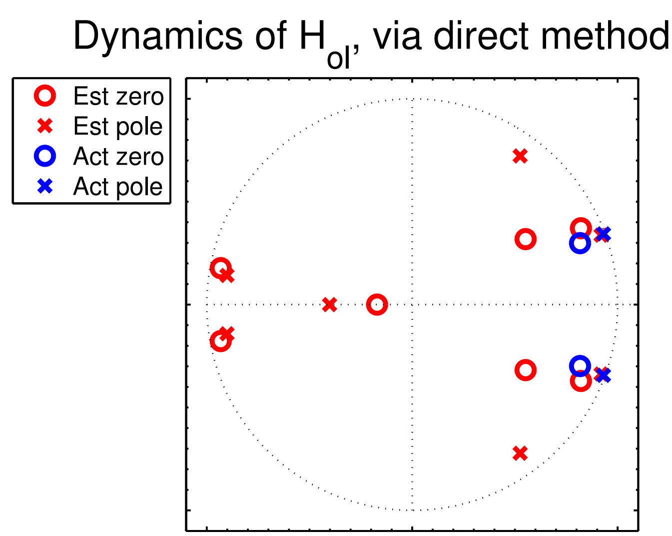






- BJ poles and zeros of $G(q)$ and $H(q)$ are reasonably well identified.

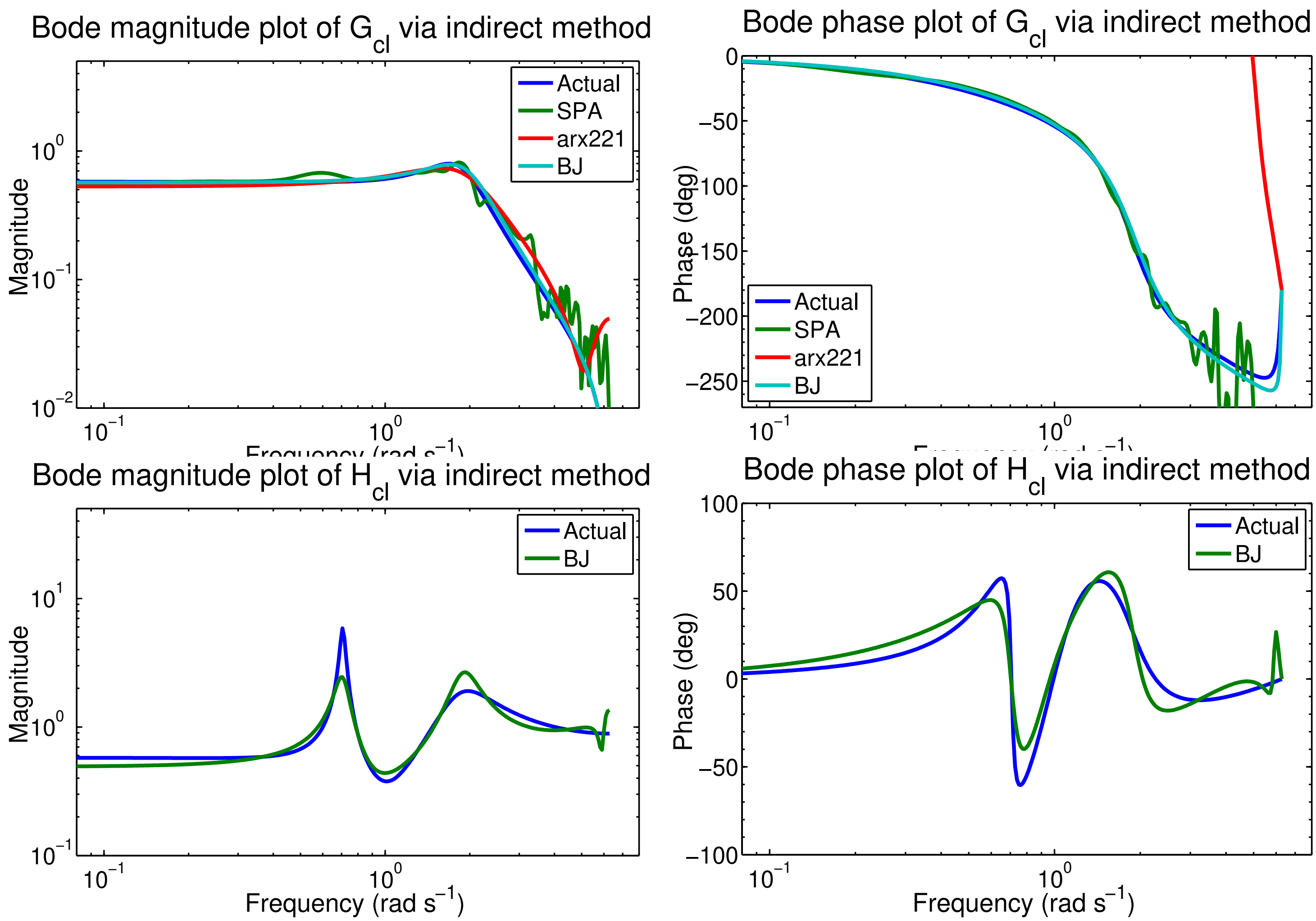




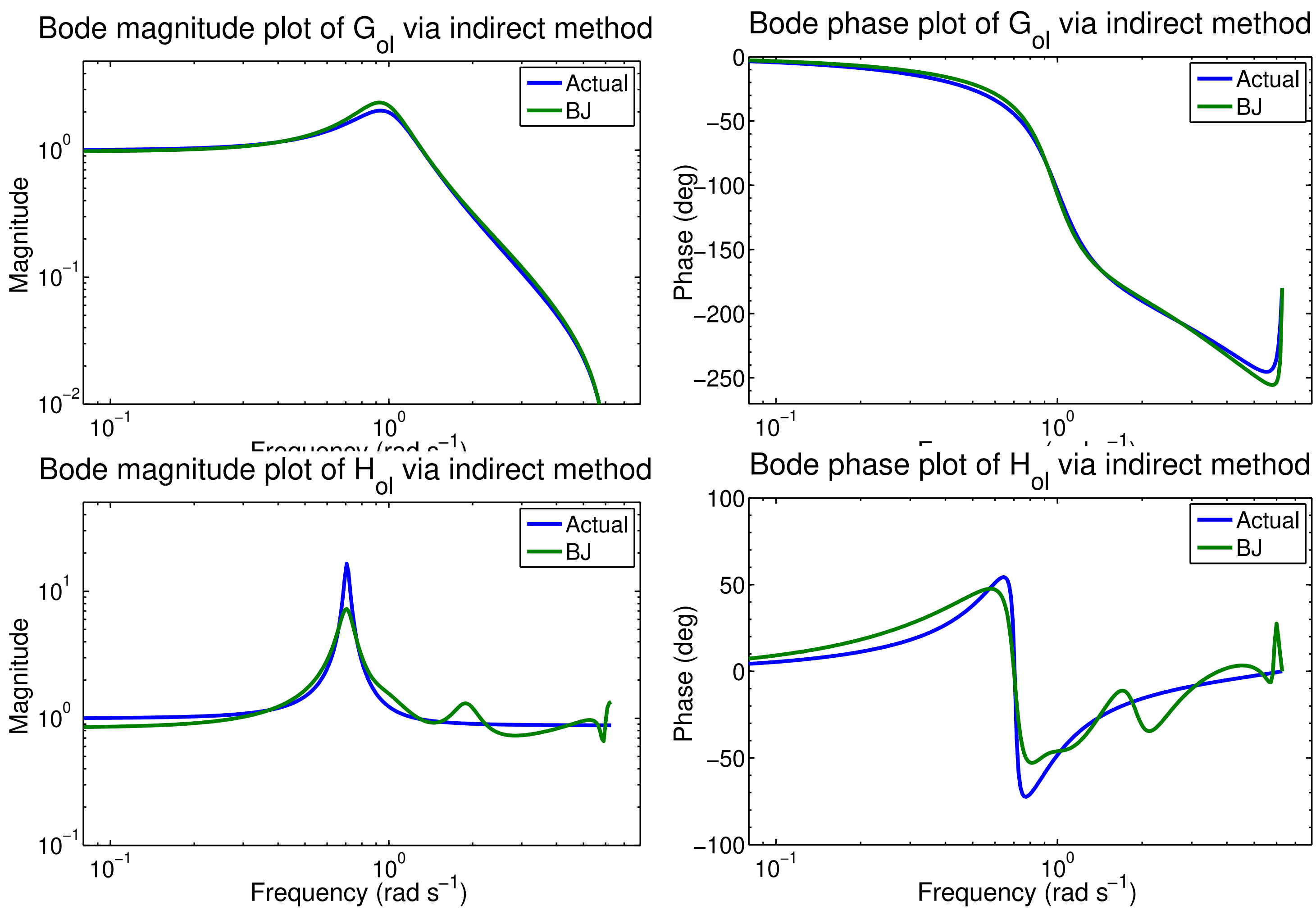
- Overall, a good fit, but not as good as we got via open-loop ID.

Indirect method

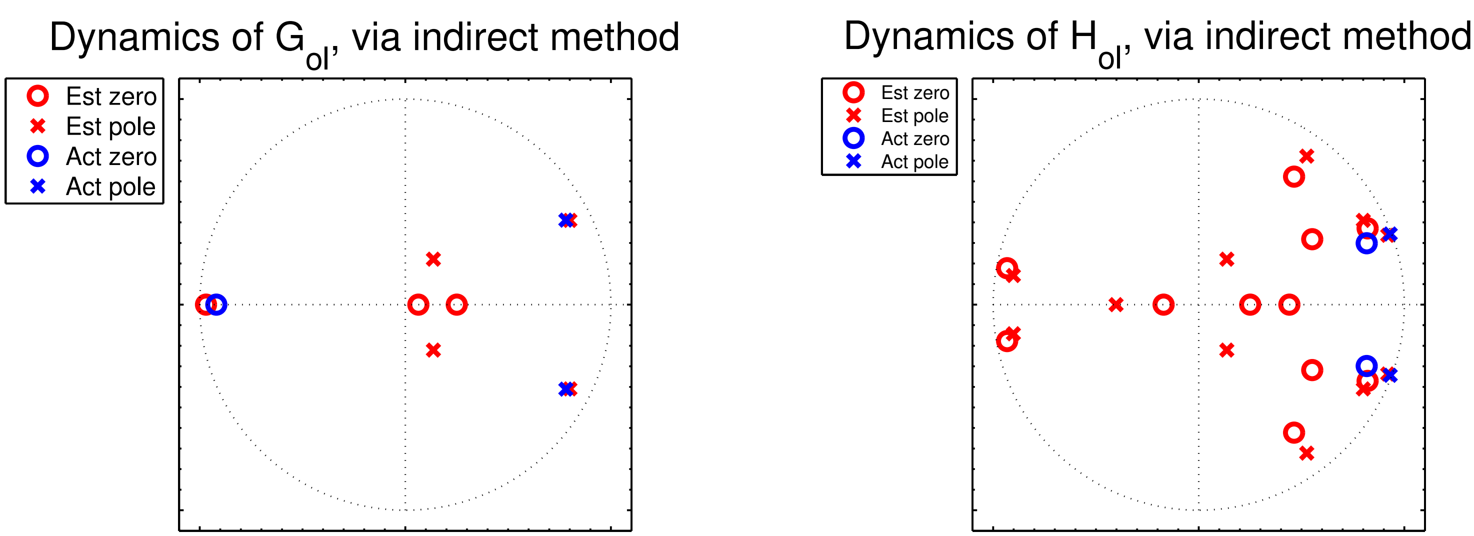
- Next, we try the indirect method.
- Identification of $\widehat{G}(q)$ and $\widehat{H}(q)$ quite good, but had to increase orders to get a decent fit: $\widehat{G}(q)$ was increased to order 331; $\widehat{H}(q)$ to 770.



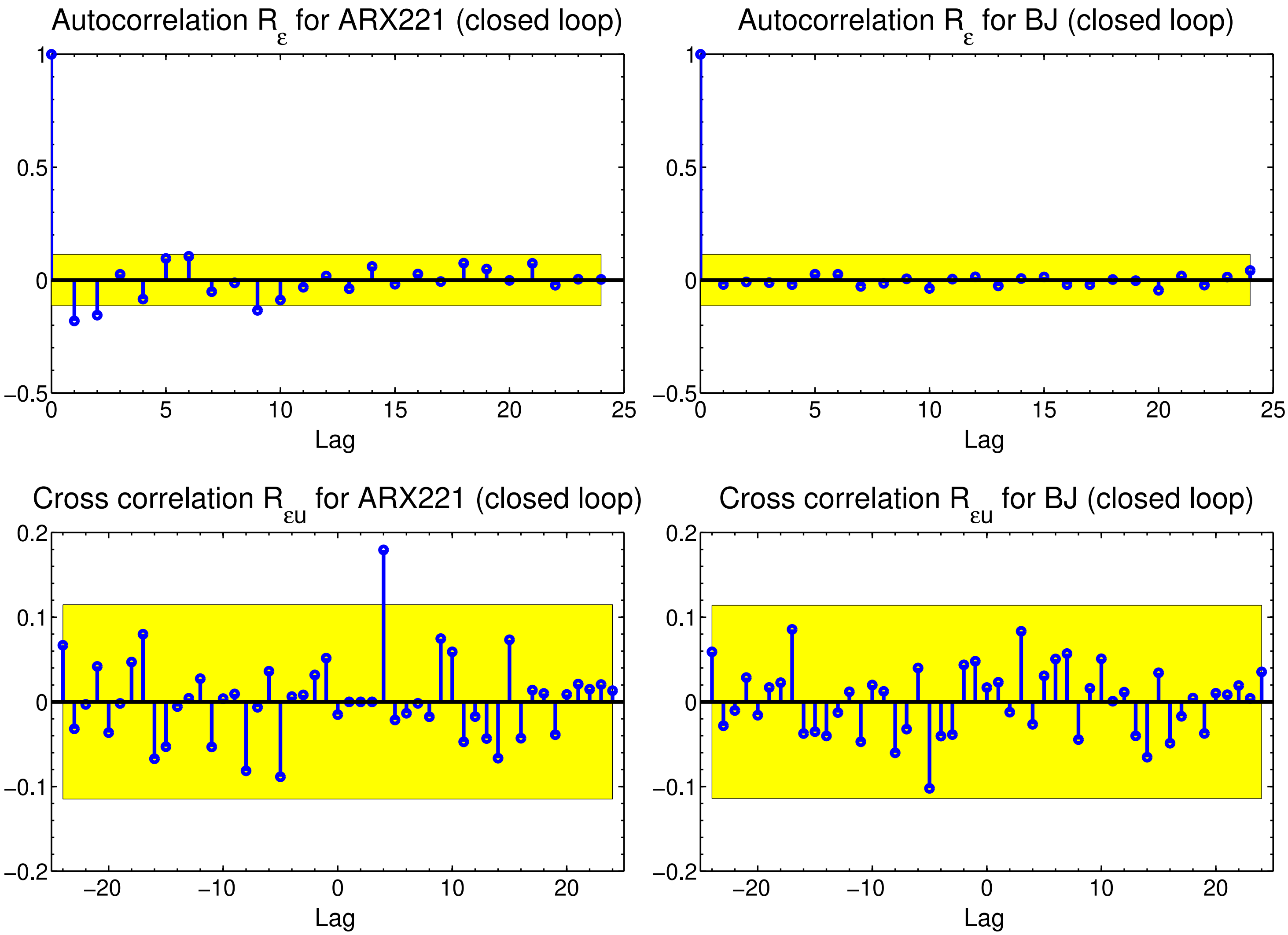
- Just to double check, the frequency responses of the open-loop estimate match the truth fairly well.



- Fit to $G(q)$ is excellent, plus some additional poles and zeros; fit to $H(q)$ is less good, plus a lot of extra poles and zeros.



- Note the near pole-zero cancellations—topic of next section.
- Residuals are much better for both methods, especially BJ.



- Before moving on, we note in passing that we have considered only closed-loop identification of transfer-function models.
- Can also ID state-space models. We don't go into details here, but:
  - Katayama has a chapter in his book, and
  - van Overschee and de Moor have also published methods: "Closed-Loop Subspace System Identification," *Proc 36th IEEE Conference on Decision and Control*, 1997.

7.3: Model-order reduction

- As we have seen, we often end up with models that are too large, and we would like to find ways to reduce the order.
- Two primary ways to do this: Truncation and residualization.

TRUNCATION: Write the system dynamics $G(q)$ given by

$$x[k+1] = Ax[k] + Bu[k]$$
$$y[k] = Cx[k] + Du[k]$$

as

$$\left[\begin{array}{c} x_1[k+1] \\ \hdashline x_2[k+1] \end{array}\right] = \left[\begin{array}{c:c} A_{11} & A_{12} \\ \hdashline A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c} x_1[k] \\ \hdashline x_2[k] \end{array}\right] + \left[\begin{array}{c} B_1 \\ \hdashline B_2 \end{array}\right] u[k]$$
$$y[k] = \left[\begin{array}{c:c} C_1 & C_2 \end{array}\right] \left[\begin{array}{c} x_1[k] \\ \hdashline x_2[k] \end{array}\right] + Du[k].$$

- We have split the states between the ones we would like to keep, $x_1[k]$, and the ones we are going to discard, $x_2[k]$.
- The truncated approximation is simply $\widehat{G}(q)$, which is given by

$$x_1[k+1] = A_{11}x_1[k] + B_1u[k]$$
$$y[k] = C_1x_1[k] + Du[k].$$

- To truncate in an intelligent way, we must organize the "most important" states into $x_1[k]$ and the "least important" states into $x_2[k]$.
- One approach is to keep slow poles (response will dominate for some time) and discard faster poles (influence dies out quickly).
  - Transform system to modal form (cf. ECE5520).

- Fast and slow poles are evident.

- Then, retain only the desired slower poles.
- Unfortunately, frequency of the pole is a poor indicator of performance. Must also determine the residue of the pole.
- Simple truncation of states can bias dc gain of the system.

RESIDUALIZATION: Tries to be more accurate than truncation.

- Approach also assumes that the $x_2[k]$ states are fast.
- These states are reduced out, but we retain the effect that their part of the model has on the lower-frequency dynamics.
- This can be accomplished by setting $x_2[k+1] = x_2[k]$, which gives

$$x_2[k] = A_{21}x_1[k] + A_{22}x_2[k] + B_2u[k]$$

$$x_2[k] = (I - A_{22})^{-1}(A_{21}x_1[k] + B_2u[k]).$$

- That is, on the time scale of the $x_1[k]$ states, the $x_2[k]$ dynamics are so fast that they are at equilibrium.
- Substituting this for $x_2[k]$ in the $x_1[k]$ equation gives:

$$\begin{aligned} x_1[k+1] &= A_{11}x_1[k] + A_{12}x_2[k] + B_1u[k] \\ &= A_{11}x_1[k] + A_{12}\left((I-A_{22})^{-1}(A_{21}x_1[k]+B_2u[k])\right) + B_1u[k] \\ &= \underbrace{\left(A_{11}+A_{12}(I-A_{22})^{-1}A_{21}\right)}_{\bar{A}} x_1[k] + \underbrace{\left(B_1+A_{12}(I-A_{22})^{-1}B_2\right)}_{\bar{B}} u[k] \\ y[k] &= C_1x_1[k] + C_2x_2[k] + Du[k] \\ &= C_1x_1[k] + C_2\left((I-A_{22})^{-1}(A_{21}x_1[k]+B_2u[k])\right) + Du[k] \\ &= \underbrace{\left(C_1+C_2(I-A_{22})^{-1}A_{21}\right)}_{\bar{C}} x_1[k] + \underbrace{\left(C_2(I-A_{22})^{-1}B_2+D\right)}_{\bar{D}} u[k]. \end{aligned}$$

- Residualization is the preferred approach because it preserves the dc gain of our model of the transfer function.
- Accuracy in both cases will depend very strongly on the importance of the eliminated states to the model of the transfer function.

Balanced realizations

- Need a good way to measure this importance. This is provided by the measures of observability and controllability.
  - Only states that are both strongly observable and controllable will contribute significantly to the system's transfer function.
  - Rank states by the product of their observability and controllability, and reduce the model size by eliminating the last ones.
- Can develop measures of the observability and controllability using the observability and controllability Gramians which are defined by the following Lyapunov equations (cf. ECE5520/5530):

$$AW_cA^T - W_c + BB^T = 0$$

$$A^TW_oA - W_o + C^TC = 0.$$

  - W_c gives an idea regarding how much energy is required to move a given state from one value to another;
  - W_o gives an idea regarding how well an ideal observer can estimate a given state in the presence of sensor noise.
- It is relatively simple to develop a similarity transformation T that yields states with

$$\widehat{W}_c = \widehat{W}_o = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n),$$

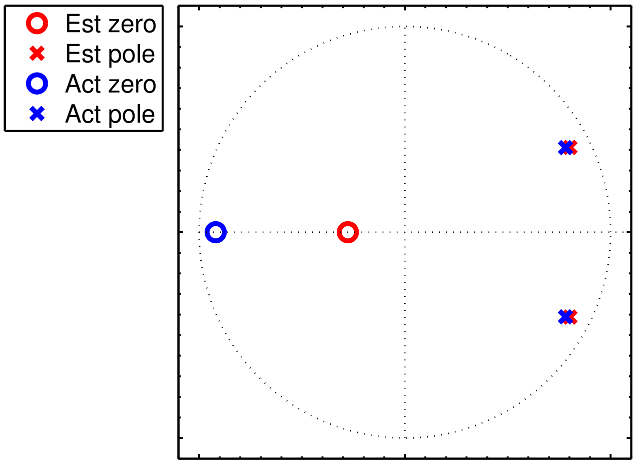
where $\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_n \ge 0$, and $\sigma_i = \sqrt{\lambda_i(\widehat{W}_c \widehat{W}_o)}$, called the balanced form of the state-space model.

- The value of σ_i provides a measure of the relative importance of the generalized state element x_i to the overall model.
- If there is a significant drop in the value of the σ's after some value of i, that would suggest that the remaining elements of the state vector are not that important.
  - Can then truncate or residualize (recommended) out these states.
- In MATLAB, use `balreal.m` and `modred.m` as in

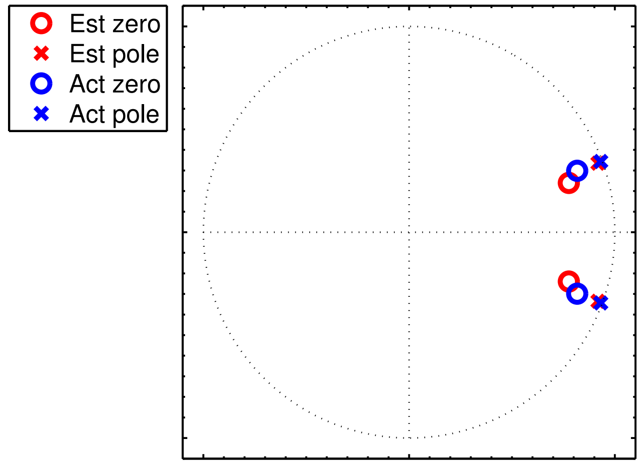
```
[sys,g] = balreal(sys)    % compute balanced realization
elim = (g<1e-8)           % small entries of g -> negligible states
rsys = modred(sys,elim)   % remove negligible states
```

- Or, use `balred.m` to do the whole operation in one step.
- Note, this will remove states associated with (actual or near) pole/zero cancellations. But, also handles MIMO systems.
- For the prior example, the reduced-order indirect transfer function estimates give the following pole-zero plots.
  - $\widehat{H}(q)$ is especially improved.

Dynamics of G_{ol}, via indirect method, red.



Dynamics of H_{ol}, via indirect method, red.



- On the numerics, you might find that C^TC and BB^T get very large.
  - Lyapunov equations in the Gramians cannot be solved.
  - Try redistributing the gain as

$$C \to C\frac{\|B\|}{\|C\|}, \qquad \text{and} \qquad B \to B\frac{\|C\|}{\|B\|},$$

  which preserves the Markov parameter CB.

- Note that VODM has an entire chapter on selecting basis functions W_1 and W_2 for subspace system identification that can aid in model-order reduction.

7.4: Recursive/real-time system identification

- We have talked about a "batch process" that uses data $y[k]$, $u[k]$, for $k = 0 \ldots N$.
- We could do this in real time by changing the "window" of past data used to calculate our estimate (*i.e.,* use the last N points).
  - Hard, but not impossible, to do the batch solution in real time.
- But, we would be much better off if we could develop a recursive update for the model estimates.
  - *i.e.,* update that requires we work on the new piece of data only.
- Various ways to do this, but let's look at the weighted least squares ARX problem

$$\hat{\theta}[N] = \arg\min_{\theta} \sum_{j=0}^{N} \lambda^{N-j} \left(y[j] - \phi^T[j]\theta\right)^2 .$$

- The term $\lambda < 1$ is a "forgetting factor" that acts to de-weight old data.
- Data appears in $y[k]$ and $\phi[k]$, parameters are in θ.
- Batch solution to this problem computes $\hat{\theta}[N] = R^{-1}[N]f[n]$, where

$$R[k] = \sum_{j=0}^{k} \lambda^{k-j}\phi[j]\phi^T[j] \quad \text{and} \quad f[k] = \sum_{j=0}^{k} \lambda^{k-j}\phi[j]y[j].$$

- When we get a new piece of data, we re-compute $R[k]$ and $f[k]$.
- But, we can do this recursively by noting that

$$R[k] = \sum_{j=0}^{k-1} \lambda^{k-j}\phi[j]\phi^T[j] + \phi[k]\phi^T[k]$$

$$= \lambda \sum_{j=0}^{k-1} \lambda^{(k-1)-j} \phi[j]\phi^T[j] + \phi[k]\phi^T[k] = \lambda R[k-1] + \phi[k]\phi^T[k],$$

and similarly

$$f[k] = \sum_{j=0}^{k-1} \lambda^{k-j} \phi[j]y[j] + \phi[k]y[k] = \lambda f[k-1] + \phi[k]y[k].$$

- This implies that

$$\begin{aligned}
\hat{\theta}[k] &= R^{-1}[k]f[k] = R^{-1}[k]\left(\lambda f[k-1] + \phi[k]y[k]\right) \\
&= R^{-1}[k]\Big(\lambda R[k-1]\hat{\theta}[k-1] + \phi[k]y[k]\Big) \\
&= R^{-1}[k]\Big[\big(R[k] - \phi[k]\phi^T[k]\big)\hat{\theta}[k-1] + \phi[k]y[k]\Big] \\
&= \hat{\theta}[k-1] + R^{-1}[k]\phi[k]\Big(y[k] - \phi^T[k]\hat{\theta}[k-1]\Big).
\end{aligned}$$

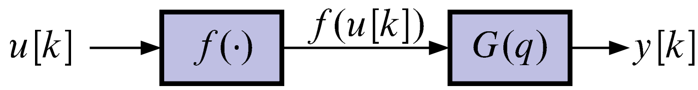
- We need to keep track of only $R[k-1]$ and $\hat{\theta}[k-1]$.
- But, we do need to invert $R[k]$ at each step. This can be avoided if we let $P[k] = R^{-1}[k]$. We can invoke the matrix inversion lemma to come up with an equivalent form:

$$\begin{aligned}
\hat{\theta}[k] &= \hat{\theta}[k-1] + L[k]\Big(y[k] - \phi^T[k]\hat{\theta}[k-1]\Big) \\
L[k] &= \frac{P[k-1]\phi[k]}{\lambda + \phi^T[k]P[k-1]\phi[k]} \\
P[k] &= \lambda^{-1}\left(P[k-1] - \frac{P[k-1]\phi[k]\phi^T[k]P[k-1]}{\lambda + \phi^T[k]P[k-1]\phi[k]}\right).
\end{aligned}$$

- Do you see the advantage?
- So we simply run these equations in real time. Still limited by ARX assumption.

7.5: Static nonlinearities

- A typical problem is that the system has a static nonlinearity on the input or output.



- Simpler than the general nonlinear case.
- If the nonlinearity is known, then we can perform standard system identification using the data set $y[k]$ and $\bar{u}[k]$, where $\bar{u}[k] = f(u[k])$.
- Usually, $f(\cdot)$ is not well known. Approximate it as a polynomial

$$f(u[k]) = \alpha_1 u[k] + \alpha_2 u^2[k] + \cdots + \alpha_m u^m[k].$$

- Pass each power of $u[k]$ through different numerator dynamics

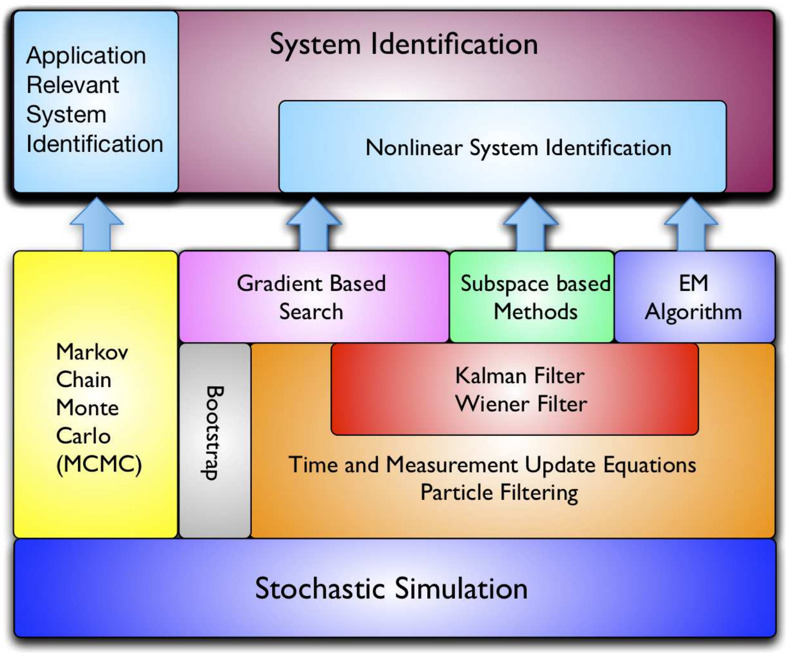
$$A(q)y[k] = B_1(q)u[k] + B_2(q)u^2[k] + \cdots + B_m(q)u^m[k].$$

- This is known as a Hammerstein model. It is simplified, but allows the use of linear system identification tools.
- For example, can use any toolbox method including transfer functions and MISO/MIMO subspace methods.

Where from here

- Hopefully you have some tools that you feel comfortable using.
- Hopefully you have some idea what the approaches are all based on:
  - PEM, based on nonlinear optimization;
  - Subspace, based on linear algebra and projections.

- Hopefully, you have done enough examples with real data that you are confident that you can model a system from data.
- As for the future, parts of the system identification field are quite mature; others are still in development.
- The graphic to the right was presented at a recent system-ID conference, showing the major tools presently employed:



- You have seen many of these in this course; *ECE5550: Applied Kalman Filtering* includes others that might be of interest.
- The same conference surveyed recent and current trends in publications in the field.
- The top categories are:
  - Nonlinear system identification;
  - Continuous-time system identification;
  - Subspace system identification;
  - Experiment design;
  - Control relevant identification.
- Best wishes as you apply these techniques to your research and future work.