A Critical View on Benchmarks based on Randomly Generated Systems

Cristian R. Rojas and Patricio E. Valenzuela
KTH Royal Institute of Technology, Sweden

Ricardo A. Rojas
Universidad Técnica Federico Santa María, Chile

ERNSI, September 20-23, 2015

Introduction

- In identification, benchmarks are essential for comparing old and new techniques to estimate models

Introduction

- In identification, benchmarks are essential for comparing old and new techniques to estimate models

- It is now customary to rely on data sets from randomly generated systems

Introduction

- In identification, benchmarks are essential for comparing old and new techniques to estimate models
- It is now customary to rely on data sets from randomly generated systems
- Here we discuss the implications of this practice, in particular when using data sets generated with the MATLAB® command `drss`

Why do we use random systems?

- Solid theoretical justification for a method not always available

Why do we use random systems?

- Solid theoretical justification for a method not always available
  - Existing bounds (Cramér-Rao / PAC) either asymptotic or too conservative

Why do we use random systems?

- Solid theoretical justification for a method not always available
  - Existing bounds (Cramér-Rao / PAC) either asymptotic or too conservative
  - ... or ignore numerical/computational issues (initialization, presence of local minima, ...)

Why do we use random systems?

- Solid theoretical justification for a method not always available
  - Existing bounds (Cramér-Rao / PAC) either asymptotic or too conservative
  - ... or ignore numerical/computational issues (initialization, presence of local minima, ...)

- Lack of good, large benchmarks of real systems

Why do we use random systems?

- Solid theoretical justification for a method not always available
  - Existing bounds (Cramér-Rao / PAC) either asymptotic or too conservative
  - ... or ignore numerical/computational issues (initialization, presence of local minima, ...)

- Lack of good, large benchmarks of real systems

- Cheap to generate random systems!

Description of drss

The MATLAB® command drss generates random discrete-time linear systems in state-space form

$$
\begin{aligned}
x_{t+1} &= Ax_t + Bu_t, \qquad u_t \in \mathbb{R}^m(\text{input}),\ x_t \in \mathbb{R}^n(\text{state}) \\
y_t &= Cx_t + Du_t, \qquad y_t \in \mathbb{R}^p(\text{output})
\end{aligned}
$$

Description of drss

The MATLAB® command `drss` generates random discrete-time linear systems in state-space form

$$
\begin{aligned}
x_{t+1} &= Ax_t + Bu_t, \qquad u_t \in \mathbb{R}^m \text{(input)},\ x_t \in \mathbb{R}^n \text{(state)} \\
y_t &= Cx_t + Du_t, \qquad y_t \in \mathbb{R}^p \text{(output)}
\end{aligned}
$$

Steps

Description of `drss`

The MATLAB® command `drss` generates random discrete-time linear systems in state-space form

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t, \qquad u_t \in \mathbb{R}^m \text{(input)},\ x_t \in \mathbb{R}^n \text{(state)} \\ y_t &= Cx_t + Du_t, \qquad y_t \in \mathbb{R}^p \text{(output)} \end{aligned}$$

Steps

1. Poles of the system are randomly selected

Description of `drss`

The MATLAB® command `drss` generates random discrete-time linear systems in state-space form

$$
\begin{aligned}
x_{t+1} &= Ax_t + Bu_t, \qquad u_t \in \mathbb{R}^m(\text{input}),\ x_t \in \mathbb{R}^n(\text{state}) \\
y_t &= Cx_t + Du_t, \qquad y_t \in \mathbb{R}^p(\text{output})
\end{aligned}
$$

Steps

1. Poles of the system are randomly selected
2. A is formed, based on the chosen poles and a random matrix of orthogonal eigenvectors

Description of `drss`

The MATLAB® command `drss` generates random discrete-time linear systems in state-space form

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t, \qquad u_t \in \mathbb{R}^m(\text{input}),\ x_t \in \mathbb{R}^n(\text{state}) \\ y_t &= Cx_t + Du_t, \qquad y_t \in \mathbb{R}^p(\text{output}) \end{aligned}$$

Steps

1. Poles of the system are randomly selected
2. A is formed, based on the chosen poles and a random matrix of orthogonal eigenvectors
3. B, C and D are generated

Description of `drss` (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system

Description of `drss` (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system
- With prob. 0.1, $p_1 \leftarrow 1$ (integrator), while $p_2, \ldots, p_n \leftarrow 1$ independently (decoupled integrators) with prob. 0.01

Description of `drss` (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system
- With prob. 0.1, $p_1 \leftarrow 1$ (integrator), while $p_2, \ldots, p_n \leftarrow 1$ independently (decoupled integrators) with prob. 0.01
- Among the non-integrator poles, grouped in pairs, each is a repeated pair of real poles with prob. 0.05

Description of `drss` (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system
- With prob. 0.1, $p_1 \leftarrow 1$ (integrator), while $p_2, \ldots, p_n \leftarrow 1$ independently (decoupled integrators) with prob. 0.01
- Among the non-integrator poles, grouped in pairs, each is a repeated pair of real poles with prob. 0.05
- The remaining poles, in pairs, are independently chosen as conjugate pairs with prob. 0.5

Description of `drss` (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system
- With prob. 0.1, $p_1 \leftarrow 1$ (integrator), while $p_2, \ldots, p_n \leftarrow 1$ independently (decoupled integrators) with prob. 0.01
- Among the non-integrator poles, grouped in pairs, each is a repeated pair of real poles with prob. 0.05
- The remaining poles, in pairs, are independently chosen as conjugate pairs with prob. 0.5
- Finally, the remaining poles are taken as distinct and real

Description of `drss` (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system
- With prob. 0.1, $p_1 \leftarrow 1$ (integrator), while $p_2, \ldots, p_n \leftarrow 1$ independently (decoupled integrators) with prob. 0.01
- Among the non-integrator poles, grouped in pairs, each is a repeated pair of real poles with prob. 0.05
- The remaining poles, in pairs, are independently chosen as conjugate pairs with prob. 0.5
- Finally, the remaining poles are taken as distinct and real

Values of poles

Description of `drss` (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system
- With prob. 0.1, $p_1 \leftarrow 1$ (integrator), while $p_2, \ldots, p_n \leftarrow 1$ independently (decoupled integrators) with prob. 0.01
- Among the non-integrator poles, grouped in pairs, each is a repeated pair of real poles with prob. 0.05
- The remaining poles, in pairs, are independently chosen as conjugate pairs with prob. 0.5
- Finally, the remaining poles are taken as distinct and real

Values of poles

- Single (non-integrator) poles and repeated poles: $\mathcal{U}[-1, 1]$

Description of drss (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system
- With prob. 0.1, $p_1 \leftarrow 1$ (integrator), while $p_2, \ldots, p_n \leftarrow 1$ independently (decoupled integrators) with prob. 0.01
- Among the non-integrator poles, grouped in pairs, each is a repeated pair of real poles with prob. 0.05
- The remaining poles, in pairs, are independently chosen as conjugate pairs with prob. 0.5
- Finally, the remaining poles are taken as distinct and real

Values of poles

- Single (non-integrator) poles and repeated poles: $\mathcal{U}[-1, 1]$
- Magnitudes of each conjugate pair: $\mathcal{U}[0, 1]$

Description of `drss` (cont.)

Generation of poles

- $p_1, \ldots, p_n$: poles of the generated system
- With prob. 0.1, $p_1 \leftarrow 1$ (integrator), while $p_2, \ldots, p_n \leftarrow 1$ independently (decoupled integrators) with prob. 0.01
- Among the non-integrator poles, grouped in pairs, each is a repeated pair of real poles with prob. 0.05
- The remaining poles, in pairs, are independently chosen as conjugate pairs with prob. 0.5
- Finally, the remaining poles are taken as distinct and real

Values of poles

- Single (non-integrator) poles and repeated poles: $\mathcal{U}[-1, 1]$
- Magnitudes of each conjugate pair: $\mathcal{U}[0, 1]$
- Arguments of each conjugate pair: $\pm\mathcal{U}[0, \pi]$

Description of `drss` (cont.)

Construction of the state matrix

$$A = U^T E U$$

Description of `drss` (cont.)

Construction of the state matrix

$$A = U^T E U$$

- $E \in \mathbb{R}^{n \times n}$: block diagonal, formed by a 1×1-block for each real pole, and a 2×2-block of the form

$$\begin{bmatrix} \mathrm{Re}(p_i) & \mathrm{Im}(p_i) \\ -\mathrm{Im}(p_i) & \mathrm{Re}(p_i) \end{bmatrix}$$

  for each conjugate pair $(p_i, \overline{p_i})$

Description of `drss` (cont.)

Construction of the state matrix

$$A = U^T E U$$

- $E \in \mathbb{R}^{n \times n}$: block diagonal, formed by a 1×1-block for each real pole, and a 2×2-block of the form

$$\begin{bmatrix} \mathrm{Re}(p_i) & \mathrm{Im}(p_i) \\ -\mathrm{Im}(p_i) & \mathrm{Re}(p_i) \end{bmatrix}$$

  for each conjugate pair $(p_i, \overline{p_i})$
- $U \in \mathbb{R}^{n \times n}$: orthogonalization of $n \times n$ $\mathcal{U}[0, 1]$ matrix

Description of `drss` (cont.)

Generation of B, C and D

$B \in \mathbb{R}^{n\times 1}$, $C \in \mathbb{R}^{1\times n}$ and $D \in \mathbb{R}^{1\times 1}$: $\mathcal{N}(0,1)$ random matrices

Description of `drss` (cont.)

Generation of B, C and D

$B \in \mathbb{R}^{n\times 1}$, $C \in \mathbb{R}^{1\times n}$ and $D \in \mathbb{R}^{1\times 1}$: $\mathcal{N}(0,1)$ random matrices

Remark In addition, `drss` zeroes some entries of B, C, D with prescribed probability

Properties of systems generated by `drss`

(i) Benchmarks of random systems induce a Bayesian comparison of identification techniques

(ii) Poles of the systems generated by `drss` do not reflect standard sampling rules-of-thumb

(iii) Effective order of systems generated by `drss` is typically much smaller than required by the user

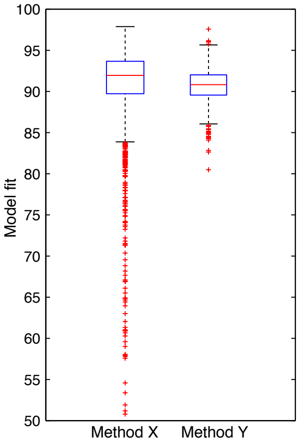
Properties of systems generated by `drss`

(i) Benchmarks of random systems induce a Bayesian comparison of identification techniques

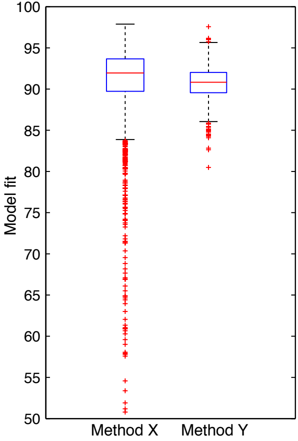
(ii) Poles of the systems generated by `drss` do not reflect standard sampling rules-of-thumb

(iii) Effective order of systems generated by `drss` is typically much smaller than required by the user

A Bayesian prior on linear systems

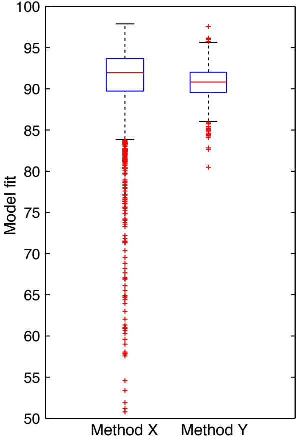


A Bayesian prior on linear systems



Which method is better?

A Bayesian prior on linear systems



Which method is better?

It depends on the distribution of the systems!

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\mathsf{FIT}}(x) = \int \underbrace{p_{\mathsf{FIT|system}}(x|s)} \; \underbrace{dP_{\mathsf{system}}(s)}$$

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\mathsf{FIT}}(x) = \int \underbrace{p_{\mathsf{FIT|system}}(x|s)}\ \underbrace{dP_{\mathsf{system}}(s)}$$

where

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\mathsf{FIT}}(x) = \int \underbrace{p_{\mathsf{FIT}|\mathsf{system}}(x|s)} \; \underbrace{dP_{\mathsf{system}}(s)}$$

where

- p_{FIT}: density function of FIT of an estimator

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\mathsf{FIT}}(x) = \int \underbrace{p_{\mathsf{FIT|system}}(x|s)}\ \underbrace{dP_{\mathsf{system}}(s)}$$

where

- p_{FIT}: density function of FIT of an estimator
- $p_{\mathsf{FIT|system}}(x|s)$: same density conditioned on system being estimated

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\mathsf{FIT}}(x) = \int \underbrace{p_{\mathsf{FIT}|\mathsf{system}}(x|s)}\ \underbrace{dP_{\mathsf{system}}(s)}$$

where

- p_{FIT}: density function of FIT of an estimator
- $p_{\mathsf{FIT}|\mathsf{system}}(x|s)$: same density conditioned on system being estimated
- P_{system}: distribution over systems in benchmark

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\mathsf{FIT}}(x) = \int \underbrace{p_{\mathsf{FIT}|\mathsf{system}}(x|s)}_{\text{estimator dependent}} \; \underbrace{dP_{\mathsf{system}}(s)}$$

where

- p_{FIT}: density function of FIT of an estimator
- $p_{\mathsf{FIT}|\mathsf{system}}(x|s)$: same density conditioned on system being estimated
- P_{system}: distribution over systems in benchmark

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\mathsf{FIT}}(x) = \int \underbrace{p_{\mathsf{FIT|system}}(x|s)}_{\text{estimator dependent}} \; \underbrace{dP_{\mathsf{system}}(s)}_{\text{given by drss!}}$$

where

- p_{FIT}: density function of FIT of an estimator
- $p_{\mathsf{FIT|system}}(x|s)$: same density conditioned on system being estimated
- P_{system}: distribution over systems in benchmark

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\mathsf{FIT}}(x) = \int \underbrace{p_{\mathsf{FIT|system}}(x|s)}_{\text{estimator dependent}} \; \underbrace{dP_{\mathsf{system}}(s)}_{\text{given by drss!}}$$

where

- p_{FIT}: density function of FIT of an estimator
- $p_{\mathsf{FIT|system}}(x|s)$: same density conditioned on system being estimated
- P_{system}: distribution over systems in benchmark

`drss` *induces a Bayesian prior over systems!*

A Bayesian prior on linear systems (cont.)

Distribution of the model fit

$$p_{\text{FIT}}(x) = \int \underbrace{p_{\text{FIT}|\text{system}}(x|s)}_{\text{estimator dependent}} \; \underbrace{dP_{\text{system}}(s)}_{\text{given by drss!}}$$

where

- p_{FIT}: density function of FIT of an estimator
- $p_{\text{FIT}|\text{system}}(x|s)$: same density conditioned on system being estimated
- P_{system}: distribution over systems in benchmark

`drss` *induces a Bayesian prior over systems!*

Is it a natural (non-informative) or realistic prior?

Properties of systems generated by `drss`

(i) Benchmarks of random systems induce a Bayesian comparison of identification techniques

(ii) Poles of the systems generated by `drss` do not reflect standard sampling rules-of-thumb

(iii) Effective order of systems generated by `drss` is typically much smaller than required by the user

Poles of generated systems

If a `drss`-generated system has n' distinct poles, their maximum magnitude is close to 1 with high probability for large n'

(the expected maximum magnitude is $n'/(n'+1)$)

Poles of generated systems (cont.)

On the other hand,

Poles of generated systems (cont.)

On the other hand,

- if real part of dominant pole of a continuous-time system is $-p^c$, the rise time is $\approx 1/p^c$, hence the sampling interval h should satisfy

$$\frac{1}{10p^c} \leq h \leq \frac{1}{4p^c}$$

Poles of generated systems (cont.)

On the other hand,

- if real part of dominant pole of a continuous-time system is $-p^c$, the rise time is $\approx 1/p^c$, hence the sampling interval h should satisfy

$$\frac{1}{10p^c} \leq h \leq \frac{1}{4p^c}$$

- Therefore, the magnitude of the sampled dominant pole, $p_{\max}$, should satisfy

$$e^{-\frac{1}{4}} \leq p_{\max} = e^{-p^c h} \leq e^{-\frac{1}{10}} \qquad \Leftrightarrow \qquad 0.78 \leq p_{\max} \leq 0.9$$

Poles of generated systems (cont.)

On the other hand,

- if real part of dominant pole of a continuous-time system is $-p^c$, the rise time is $\approx 1/p^c$, hence the sampling interval h should satisfy

$$\frac{1}{10p^c} \leq h \leq \frac{1}{4p^c}$$

- Therefore, the magnitude of the sampled dominant pole, $p_{\max}$, should satisfy

$$e^{-\frac{1}{4}} \leq p_{\max} = e^{-p^c h} \leq e^{-\frac{1}{10}} \qquad \Leftrightarrow \qquad 0.78 \leq p_{\max} \leq 0.9$$

- $\Rightarrow$ `drss` can generate, for large n, the equivalents of severely over-sampled systems

Poles of generated systems (cont.)

- Over-sampled systems can give rise to several numerical problems. Special techniques have been developed for handling these issues (*e.g.*, delta operator), but

Poles of generated systems (cont.)

- Over-sampled systems can give rise to several numerical problems. Special techniques have been developed for handling these issues (*e.g.*, delta operator), but

  should these systems be used for comparing general-purpose estimators?

Poles of generated systems (cont.)

- Over-sampled systems can give rise to several numerical problems. Special techniques have been developed for handling these issues (*e.g.*, delta operator), but

  should these systems be used for comparing general-purpose estimators?

- Similarly, random systems with dominant poles of small magnitude correspond to under-sampled systems, whose estimation can be difficult (poor observability/identifiability)

Poles of generated systems (cont.)

- Over-sampled systems can give rise to several numerical problems. Special techniques have been developed for handling these issues (*e.g.*, delta operator), but

  should these systems be used for comparing general-purpose estimators?

- Similarly, random systems with dominant poles of small magnitude correspond to under-sampled systems, whose estimation can be difficult (poor observability/identifiability)

- In summary: **poles of random systems should be carefully placed to represent how sampled systems would look like** (assuming sampling and experiment design are properly done)

Properties of systems generated by `drss`

(i) Benchmarks of random systems induce a Bayesian comparison of identification techniques

(ii) Poles of the systems generated by `drss` do not reflect standard sampling rules-of-thumb

(iii) Effective order of systems generated by `drss` is typically much smaller than required by the user

Low order random systems

- There are many ways to define the "effective order" of a system G

Low order random systems

- There are many ways to define the "effective order" of a system G
- Here we use

$$\text{eff}(G, a) := \#\{\sigma_i \geq a\sigma_1 : \ 1 \leq i \leq n\}$$

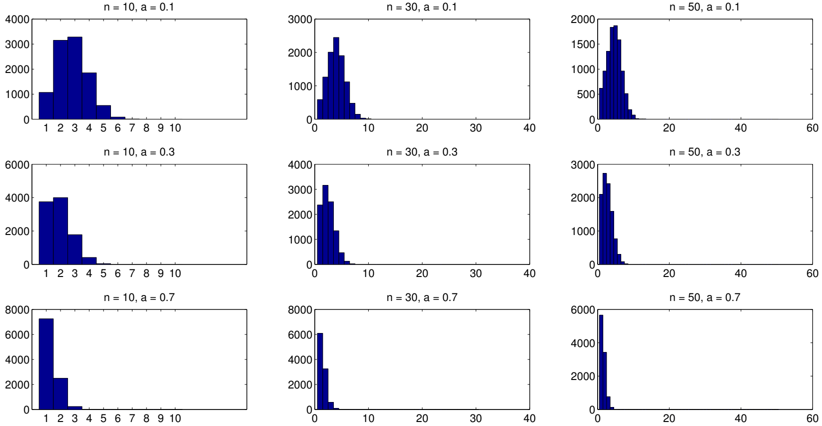
Low order random systems

- There are many ways to define the "effective order" of a system G
- Here we use

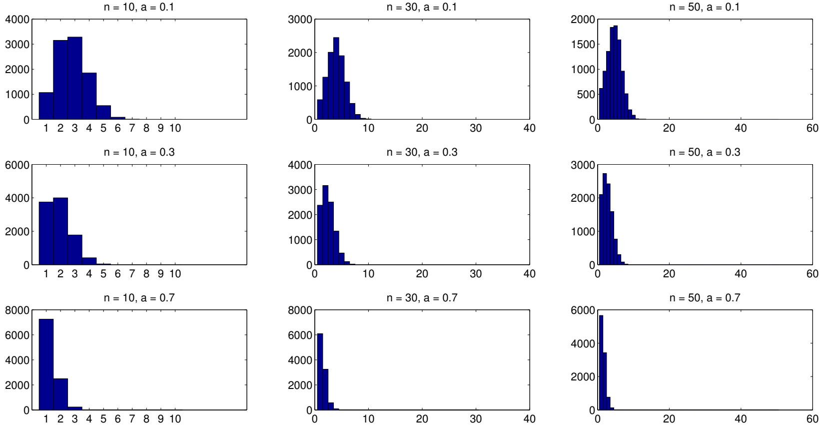
$$\text{eff}(G, a) := \#\{\sigma_i \geq a\sigma_1 : \ 1 \leq i \leq n\}$$

where
$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$: Hankel singular values of G
a: threshold on number of significant Hankel singular values

Low order random systems (cont.)

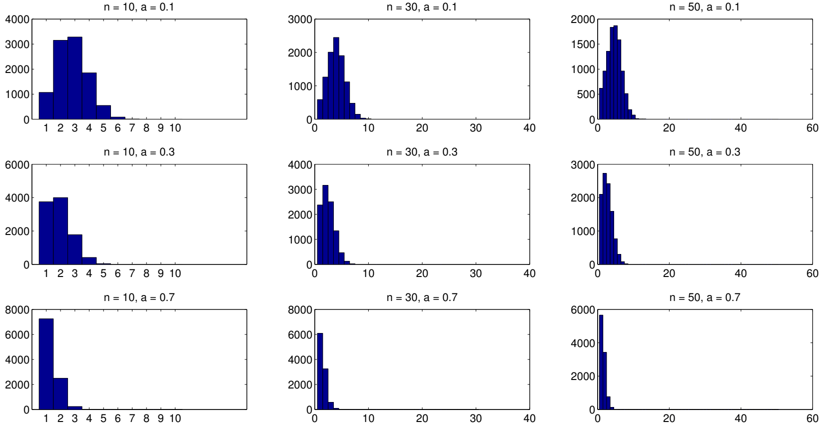


Low order random systems (cont.)



`drss` tends to generate systems with very low effective order!

Low order random systems (cont.)



`drss` tends to generate systems with very low effective order!

is this good or bad?

Some suggestions

- Partition the benchmark systems into subsets
- Plot the joint distribution of performance measures
- Sample randomly generated continuous-time systems
- Try "irreducible" systems

Partition the benchmark into subsets

- A prior prioritizes some classes of systems over others, not necessarily in agreement with the real industrial practice

Partition the benchmark into subsets

- A prior prioritizes some classes of systems over others, not necessarily in agreement with the real industrial practice

- An alternative is to split the set of generated systems into several subclasses (with similar dynamics, order, resonances, *etc.*), and test the estimators on each subclass *separately*

Partition the benchmark into subsets

- A prior prioritizes some classes of systems over others, not necessarily in agreement with the real industrial practice
- An alternative is to split the set of generated systems into several subclasses (with similar dynamics, order, resonances, *etc.*), and test the estimators on each subclass *separately*
- This would allow to distinguish conditions under which an estimator outperforms others

Plot the joint distribution of the FIT/MSE

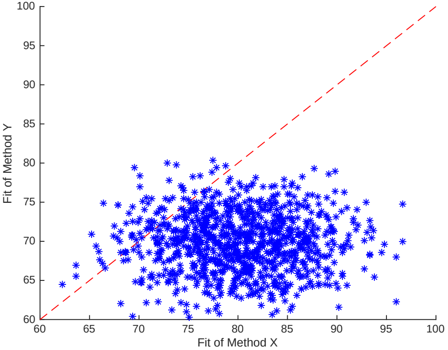
- There is more information in the results of using benchmarks than that contained in box plots (marginal FIT distributions)

Plot the joint distribution of the FIT/MSE

- There is more information in the results of using benchmarks than that contained in box plots (marginal FIT distributions)
- Instead, consider presenting the joint FIT distribution:

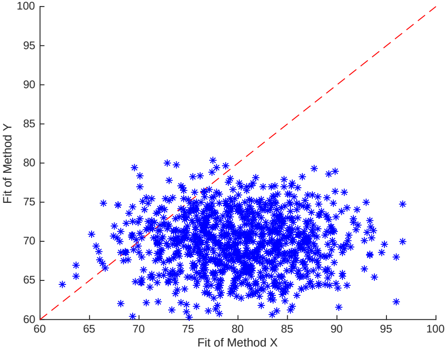
Plot the joint distribution of the FIT/MSE

- There is more information in the results of using benchmarks than that contained in box plots (marginal FIT distributions)
- Instead, consider presenting the joint FIT distribution:



Plot the joint distribution of the FIT/MSE

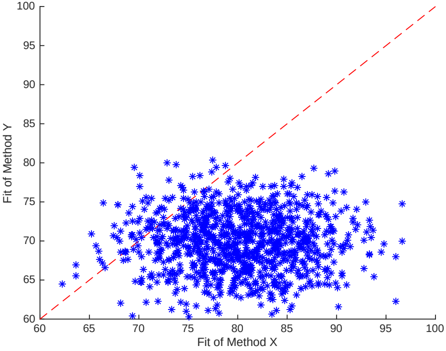
- There is more information in the results of using benchmarks than that contained in box plots (marginal FIT distributions)
- Instead, consider presenting the joint FIT distribution:



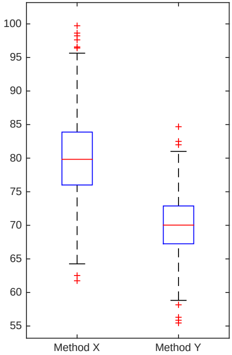
vs.

Plot the joint distribution of the FIT/MSE

- There is more information in the results of using benchmarks than that contained in box plots (marginal FIT distributions)
- Instead, consider presenting the joint FIT distribution:



vs.



Sample random continuous-time systems

- To generate systems whose poles follow sampling rules-of-thumb, one can start with a random continuous-time system and then sample it!
  (as in the d(1/2)s(1/2) benchmark of Chen,Ohlsson&Ljung)

Sample random continuous-time systems

- To generate systems whose poles follow sampling rules-of-thumb, one can start with a random continuous-time system and then sample it!
  (as in the d(1/2)s(1/2) benchmark of Chen,Ohlsson&Ljung)

- This procedure would not only lead to an improved choice of poles, but also of the zeros of the generated system, because it would introduce the right sampling zeros

Sample random continuous-time systems

- To generate systems whose poles follow sampling rules-of-thumb, one can start with a random continuous-time system and then sample it!
  (as in the d(1/2)s(1/2) benchmark of Chen,Ohlsson&Ljung)

- This procedure would not only lead to an improved choice of poles, but also of the zeros of the generated system, because it would introduce the right sampling zeros

- Which prior to use in continuous-time?

“Irreducible” systems

- To test estimators on systems of high effective order, we need systems most of whose Hankel singular values are significant

"Irreducible" systems

- To test estimators on systems of high effective order, we need systems most of whose Hankel singular values are significant

- All-pass systems are hard to reduce, in the sense that

$$H(z) = K \frac{(1 - \overline{p_1} z) \cdots (1 - \overline{p_n} z)}{(z - p_1) \cdots (z - p_n)}$$

have all their Hankel singular values equal!

“Irreducible” systems

- To test estimators on systems of high effective order, we need systems most of whose Hankel singular values are significant

- All-pass systems are hard to reduce, in the sense that

$$H(z) = K\frac{(1-\overline{p_1}z)\cdots(1-\overline{p_n}z)}{(z-p_1)\cdots(z-p_n)}$$

  have all their Hankel singular values equal!

- These systems may not be *realistic*, but may serve to test estimators on problem of real high order

Conclusions

- Designing a good benchmark of artificial systems is difficult

Conclusions

- Designing a good benchmark of artificial systems is difficult
- Some suggestions, but mostly more questions

Conclusions

- Designing a good benchmark of artificial systems is difficult
- Some suggestions, but mostly more questions
- As we rely more and more on benchmark comparisons, their design and proper use should be seriously studied:

Conclusions

- Designing a good benchmark of artificial systems is difficult
- Some suggestions, but mostly more questions
- As we rely more and more on benchmark comparisons, their design and proper use should be seriously studied:
  - Classes of systems to consider?

Conclusions

- Designing a good benchmark of artificial systems is difficult
- Some suggestions, but mostly more questions
- As we rely more and more on benchmark comparisons, their design and proper use should be seriously studied:
  - Classes of systems to consider?
  - What are we comparing: as benchmark tests depend on not only statistical, but also numerical (conditioning) and computational (initial condition, local minima) issues, they are heavily implementation-dependent

Conclusions

- Designing a good benchmark of artificial systems is difficult
- Some suggestions, but mostly more questions
- As we rely more and more on benchmark comparisons, their design and proper use should be seriously studied:
  - Classes of systems to consider?
  - What are we comparing: as benchmark tests depend on not only statistical, but also numerical (conditioning) and computational (initial condition, local minima) issues, they are heavily implementation-dependent
  - How should we present the results of Monte Carlo studies?

Thank you!