Estimation of Transfer Function from Bode Plot Images Using Multimodal Large Language Model

Shunya Hirano[1*] and Masanobu Koga[2]

[1]Department of Creative Informatics, Graduate School of Computer Science and Systems Engineering, Kyushu Institute of Technology, Japan (hirano.shunya804@mail.kyutech.jp) * Corresponding author
[2]Department of Intelligent and Control Systems, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Japan (koga@ics.kyutech.ac.jp)

Abstract: Toward the realization of control design support using Multimodal Large Language Model (MLLM), this study proposes a method in which the MLLM estimates a transfer function from a Bode plot image. The proposed method consists of two stages. In the first stage, the MLLM extracts frequency-response data points from a Bode plot image. In the second stage, the extracted data are provided to another LLM to estimate a transfer function and to evaluate the errors with respect to the input data. Using the same points extracted in the first stage, we also compare the estimation by MATLAB. The experiments showed that direct estimation from an image achieved high accuracy, whereas the two-stage method further reduced the estimation error for systems with closely spaced poles and zeros, nonminimum-phase systems, and high-order systems. The orders of estimated transfer function by the LLM often matched the true orders, whereas the orders of estimated transfer function by MATLAB tended to be higher than the true orders. Although the two-stage method increases the computational cost, the accuracy could be improved for the transfer functions with complex structures, because the search uses the explicit intermediate representation and the self-evaluation of fitting errors.

Keywords: MLLM, LLM, Bode plot, transfer function estimation

1. INTRODUCTION

In recent years, multimodal large language models (MLLMs) have achieved remarkable development, and can process not only text but also images, audio, and video in an integrated manner[1].

Research on extracting numerical information and structured representations from graph images using MLLMs has also progressed[2], [3]. In these studies, restoring the numerical values and the structures contained in graphs as JSON, plotting code, or the like has been investigated.

In control design, graphs such as frequency responses and time responses are used for the analysis of system characteristics. In particular, stability margins, bandwidth, poles, zeros, and the like can be obtained from a Bode plot, and are used for design decisions[4]. Therefore, in order to utilize LLMs for control support, an LLM must read numerical values accurately from these graphs. In addition, in order to perform control system design and tuning, the ability to grasp the characteristics of the target system from the numerical values read from the graphs is also necessary. However, the studies in [2], [3] do not address using the information obtained from the graph to estimate a model of the system that the graph represents.

In this study, toward the realization of control design support using an MLLM, we propose a method in which the MLLM estimates a transfer function from a Bode plot image. The proposed method consists of two stages.

Furthermore, using the frequency-response data extracted in the first stage as the input, we perform identification using MATLAB's `tfest` function and `patternsearch`, and compare its result with that obtained by the proposed two-stage method. From the viewpoints of the estimation accuracy, the model structure, the computational cost, and the error propagation, we clarify the estimation characteristics of the LLM and MATLAB.

This paper is organized as follows. Section 2 describes the proposed method, Section 3 describes the experiments, Section 4 presents the results and discussion, and Section 5 presents the conclusion. In this paper, models that take only text as input and models that also support image input are collectively denoted as LLM, and MLLM is used when explicitly referring to processing that involves image input.

2. PROPOSED METHOD

2.1 Overview of the Proposed Method

Figure 1 shows the flow of the proposed method. The proposed method consists of two stages. In the first stage, the MLLM extracts frequency-response data consisting of the angular frequency, the gain, and the phase from a Bode plot image. In the second stage, the extracted frequency-response data are provided to the LLM, and a transfer function is estimated. Furthermore, the LLM itself is instructed to calculate the errors between the estimated transfer function and the input frequency-response data. In this configuration, the LLM can numerically evaluate its own estimation result against the explicit intermediate representation.

The first stage and the second stage are performed as independent LLM calls. The extracted points output by the first stage are saved as a CSV file and attached to the input of the second stage. No conversation history is carried over between the two. Therefore, the second stage estimates a transfer function only from the extracted

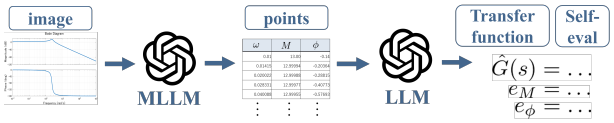

Fig. 1. Proposed method.

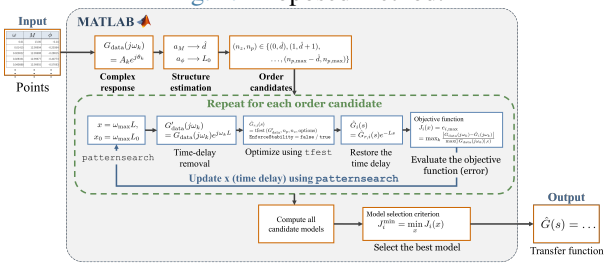

Fig. 2. MATLAB-based transfer-function identification procedure with model-order and time-delay search.

points, without referring to the Bode plot image or the reasoning process of the first stage.

2.2 Point Extraction

In the first stage, a Bode plot image is provided to the MLLM, and the MLLM is instructed to output frequency-response data consisting of N extracted points,

$$\{(\omega_k, M_k, \phi_k)\}_{k=1}^{N}.$$

Here, ω_k denotes the angular frequency [rad/s], M_k the gain [dB], and ϕ_k the unwrapped phase [deg].

2.3 Transfer-Function Estimation

In the second stage, the frequency-response data extracted in the first stage are provided to the LLM. The estimated transfer function is expressed as

$$\hat{G}(s) = \frac{b_{n_z} s^{n_z} + b_{n_z-1} s^{n_z-1} + \cdots + b_0}{a_{n_p} s^{n_p} + a_{n_p-1} s^{n_p-1} + \cdots + a_0} e^{-Ls}.$$

The LLM is instructed to generate the coefficient vectors

$$\text{num} = [b_{n_z}, b_{n_z-1}, \ldots, b_0], \ \text{den} = [a_{n_p}, a_{n_p-1}, \ldots, a_0],$$

that constitute this transfer function, and the time delay L. Here, n_z and n_p are the order of the numerator and the order of the denominator, respectively. Furthermore, for the estimated transfer function and the input frequency-response data, the LLM is instructed to generate the maximum absolute gain error e_M [dB] and the maximum absolute phase error e_ϕ [deg].

2.4 MATLAB Identification as a Comparison Method

This section describes MATLAB identification that uses the frequency-response data extracted in the first stage of the proposed method. The frequency-response data extracted in the first stage are input to MATLAB[5], and the transfer function $G(s)$ is obtained by optimization. Figure 2 shows the flow of the procedure for the transfer-function estimation.

The complex frequency response based on the frequency-response data obtained in the first stage is expressed as

$$G_{\text{data}}(j\omega_k) = A_k e^{j\theta_k},$$

where

$$A_k = 10^{M_k/20}, \quad \theta_k = \frac{\pi}{180}\phi_k.$$

In the high-frequency region, the gain slope a_M [dB/dec] with respect to the common logarithm of the angular frequency is obtained. From a_M, the relative degree of the transfer function is obtained by

$$\hat{d} = \max\left(0, \text{round}\left(-\frac{a_M}{20}\right)\right),$$

where round is a function that rounds a value to the nearest integer.

In addition, in the high-frequency region, the phase slope a_ϕ [s] with respect to the angular frequency is obtained. From a_ϕ, the initial value of the estimated time delay

$$\hat{L}_0 = -a_\phi$$

is obtained.

To preserve the relative degree $\hat{d}$, the set of candidate (n_z, n_p)

$$S = \{(0, \hat{d}), (1, \hat{d}+1), ..., (n_{p,\max} - \hat{d}, n_{p,\max})\}$$

is defined.

For each order candidate $s_i \in S (i = 1, \ldots, n_{p,\max})$, the estimated time delay $\hat{L}_m$ is searched for by MATLAB's `patternsearch`, where

$$x_m = \omega_{\max} \hat{L}_m$$

is used as search variable. Here, $\omega_{\max}$ denotes the upper limit of the angular frequency range displayed in the Bode plot image. Using the estimated time delay $\hat{L}_m$,

$$G'_{\text{data}}(j\omega_k) = G_{\text{data}}(j\omega_k) e^{j\omega_k \hat{L}_m},$$

is generated, where the time-delay component has been removed.

The candidate of the transfer function excluding the time delay is defined as

$$\hat{G}_{\text{r,i}}(s) = \frac{b_{i,n_z} s^{n_z} + b_{i,n_z-1} s^{n_z-1} + \cdots + b_{i,0}}{a_{i,n_p} s^{n_p} + a_{i,n_p-1} s^{n_p-1} + \cdots + a_{i,0}}.$$

For each candidate, the complex frequency-response data $\{\omega_k, G'_{\text{data}}(j\omega_k)\}_{k=1}^{N}$, n_p, n_z, and the estimation options are input to MATLAB's `tfest` [6], [7], and a transfer-function model with the specified structure is estimated.

With the time delay $\hat{L}_m$ fixed, `tfest` estimates the coefficients for $G'_{\text{data}}(j\omega_k)$. After the estimation, the time delay is restored to the model.

$$\hat{G}_i(s) = \hat{G}_{\text{r,i}}(s) e^{-\hat{L}_m s}$$

The obtained model is evaluated using the maximum complex relative error at the N input points

$$J_i(x) = e_{i,\max}(x) = \max_{1 \le k \le N} \frac{\left| G_{\text{data}}(j\omega_k) - \hat{G}_i(j\omega_k) \right|}{\max\left(\left| G_{\text{data}}(j\omega_k) \right|, \epsilon \right)}.$$

Table 1. Comparison methods used in the experiments.

Method	Input	Intermediate	Estimator	Stability
Direct MLLM	Image	None	MLLM	–
Two-stage LLM	Image	Frequency-response points	LLM	–
LLM+MATLAB-N	Image	Frequency-response points	MATLAB	None
LLM+MATLAB-S	Image	Frequency-response points	MATLAB	Enforced

Here, ϵ is a small positive value for preventing division by zero.

`patternsearch` performs the search using $J_i(x)$ as the objective function until the termination condition is satisfied.

For each order candidate s_i, the minimum objective value obtained by the search is

$$J_i^{\min} = \min_x J_i(x).$$

The model with the smallest $J_i^{\min}$ is selected.

3. EXPERIMENTS

This section describes the experiments for confirming the effectiveness of the proposed method. Table 1 and Figure 3 show the configuration of each estimation method.

In Direct MLLM, a Bode plot image is provided directly to the MLLM, and a transfer function is estimated.

In Two-stage LLM, the proposed method described in Section 2 is used. In addition, the number of extracted points was set to $N = 200$, and the LLM was instructed to generate them in ascending order of ω_k.

In MATLAB, a transfer function is estimated using the same frequency-response data as those of Two-stage LLM. This makes it possible, with the point-extraction conditions kept common, to compare the two-stage End-to-End estimation for the case in which an LLM is used as the second-stage estimator and for the case in which MATLAB identification is used. Moreover, in the identification by MATLAB, two conditions are evaluated: without a stability constraint (MATLAB-N) and with a stability constraint (MATLAB-S). In addition, the maximum denominator order searched by MATLAB was set to $n_{\mathrm{p,max}} = 10$. $G_{\mathrm{data}}(j\omega_k)$, obtained by converting the frequency-response data extracted by the LLM into a complex response, was used as the input. As the estimation option, `WeightingFilter="inv"` was set in order to emphasize the relative fit of the frequency response.

3.1 Experimental Conditions

Table 2 shows the LLM API settings, the experimental environment, and the libraries used in this experiment. The settings were common to each call of Direct MLLM and Two-stage LLM. Code Interpreter is an OpenAI tool that allows a model to generate and execute Python code in a sandbox environment and to use the execution results in its reasoning[8]. In this experiment, the use of this tool

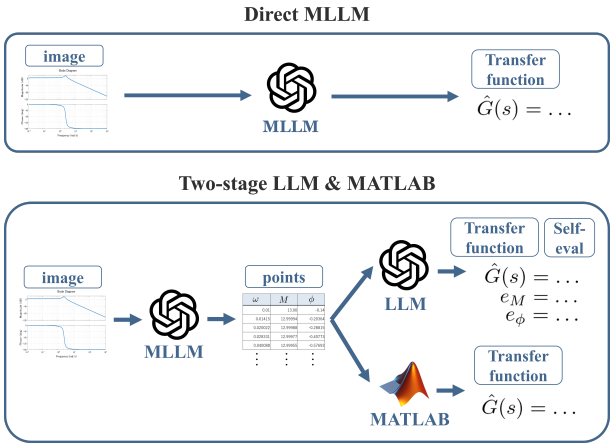

Fig. 3. Experimental methods.

Table 2. Experimental environment and API settings.

Item	Value
LLM model	GPT-5.6 Sol (gpt-5.6-sol)
LLM output format	JSON Schema
LLM instructions	None
Image detail level	Original
Reasoning effort	Medium
Reasoning mode	Standard
Service tier	Default
Prompt language	English
LLM tool	Code Interpreter
Code Interpreter memory	4 GB
Programming language	Python 3.13.13
MATLAB	R2026a Update 2
Host OS	Windows 11
Host CPU	Intel Core Ultra 7 258V
Host RAM	32 GB

was required for all LLM calls, and the executed code and outputs were recorded.

In this experiment, six transfer functions with various complexities were used as the test systems. Table 3 shows their configurations. The corresponding transfer functions are provided in the Appendix. The Bode plot image of each transfer function was generated with MATLAB and used as the input to each estimation method.

Table 4 shows the generation conditions of the Bode plot images and the common experimental conditions. The representative frequencies of the pole and zero elements were placed between 10^{-1} and 10^{2} rad/s, and the displayed frequency range was set so that their effects could be observed.

For each test system, the estimation was executed three times. However, for G6, the trial, in which the point extraction broke down substantially once, was re-executed, and the three trials that completed normally were used for the evaluation.

Figure 4 shows the prompts used in this experiment. The prompts were determined in advance using the system different from the test systems. They describe only the explanation of the input information and the instructions and constraints for the output, and do not specify concrete analysis methods or processing procedures. The output format was specified as structured output conforming to a JSON Schema.

Table 3. Transfer functions used in the experiments.

System	Main characteristic	FO poles	SO pole pairs	FO zeros	SO zero pairs	Special property
G1	Reference system with first- and second-order elements	2	1	1	1	3rd/4th order
G2	Closely spaced first-order poles and zeros	5	0	2	0	Close elements
G3	Second-order elements with high damping ratios	0	2	0	1	High damping
G4	Same rational part as G1 with a time delay	2	1	1	1	$L = 0.01$ s
G5	Nonminimum-phase system based on G1	2	1	1	1	One RHP zero
G6	High-order system with multiple first- and second-order elements	4	2	2	2	6th/8th order

FO and SO denote first-order and second-order elements, respectively. Each SO pole or zero pair denotes one complex-conjugate pair.

Table 4. Common experimental conditions.

Item	Value
Image size	1412×955 px
Resolution	200 dpi
Observed curve width	Approx. 2 px
Frequency range	10^{-2}–10^{3} rad/s
Trials per system and method	3

Direct MLLM
Estimate a transfer function G(s), from the attached Bode plot image.

Two-stage LLM (Stage 1)
Extract 200 frequency-response data points, from the attached Bode plot image.
- Represent each point as a tuple (angular frequency, gain, phase).
- Sort the points in ascending order of angular frequency.
- Unwrap the phase to make it continuous.

Two-stage LLM (Stage 2)
The attached file contains frequency-response data extracted from a Bode plot. Each data point is a tuple (angular frequency, gain, unwrapped phase). Estimate a transfer function (G(s)) from these frequency-response data. Also, calculate and output the maximum absolute errors in gain and phase between the estimated transfer function and the input frequency-response data.

Fig. 4. Prompts used in the experiments.

3.2 Evaluation Metrics

3.2.1 End-to-End Frequency-Response Accuracy

The End-to-End frequency-response accuracy is evaluated by placing 10,000 points in the range between 10^{-2} and 10^{3} rad/s so that $\log_{10}\omega$ is equally spaced, and comparing the true transfer function with the estimated transfer function. As the evaluation metrics, the maximum complex relative error and the maximum absolute gain and phase errors are used.

$$e_{\mathrm{c,max}} = \max_k \frac{\left|G_{\mathrm{true}}(j\omega_k) - \hat{G}(j\omega_k)\right|}{(|G_{\mathrm{true}}(j\omega_k)|)},$$

$$e_{M,\mathrm{max}} = \max_k \left|M_{\mathrm{true}}(\omega_k) - \hat{M}(\omega_k)\right|,$$

$$e_{\phi,\mathrm{max}} = \max_k \left|\phi_{\mathrm{true}}(\omega_k) - \hat{\phi}(\omega_k)\right|$$

3.2.2 Transfer-Function Structure Estimation

For each test system, the numerator order and the denominator order of the estimated transfer function are compared with their true values.

For G4, which includes a time delay, the absolute error

$$e_L = |L_{\mathrm{true}} - \hat{L}|$$

between the true time delay L_{true} and the estimated time delay $\hat{L}$ is evaluated.

For G5, which includes a right-half-plane (RHP) zero, the number of RHP zeros in the estimated transfer function is compared with the true value, and the distance between the true RHP zero and the closest estimated zero is also evaluated.

For the identification by MATLAB, the stability is determined on the basis of the real parts of the estimated poles, and the structural differences due to the presence or absence of the stability constraint are evaluated.

3.2.3 Computational Cost

The computational cost is evaluated in terms of the processing time, the API token usage, and the API fee of each method.

3.2.4 Error Propagation

We evaluate how the error of the extracted frequency-response data propagates as a result of the subsequent transfer-function estimation. In this evaluation, the 200 frequency points extracted by the MLLM are used. Let e_{ext} denote the maximum error at the point extraction, e_{LLM} the final estimation error by Two-stage LLM, e_{N} the final estimation error by MATLAB-N, and e_{S} the final estimation error by MATLAB-S. The error amplification factor of each estimation method is defined by

$$\kappa_{\mathrm{LLM}} = \frac{e_{\mathrm{LLM}}}{e_{\mathrm{ext}}}, \quad \kappa_{\mathrm{N}} = \frac{e_{\mathrm{N}}}{e_{\mathrm{ext}}}, \quad \kappa_{\mathrm{S}} = \frac{e_{\mathrm{S}}}{e_{\mathrm{ext}}}$$

A value of $\kappa < 1$ indicates that the error at the point extraction was reduced by the subsequent estimation, whereas $\kappa > 1$ indicates that it was amplified.

4. RESULTS AND DISCUSSION

4.1 End-to-End Frequency-Response Accuracy

Table 5 shows the mean values of the maximum End-to-End frequency-response errors over the three trials of each method. Although small errors were obtained for all test systems with Direct MLLM, Two-stage LLM showed smaller errors than Direct MLLM in all trials for G2, G5, and G6. In contrast, Direct MLLM was superior for G1. The Code Interpreter logs showed that, in multiple Two-stage LLM runs, the error was first calculated with respect to the input data, after which the candidate structure or the model order was changed, re-optimization was performed, and the final error was checked again. This suggests that the self-evaluation was used for further exploration of candidate models. Note that Direct MLLM

Table 5. Mean End-to-End errors over three trials.

Sys	Method	$e_{c,\max}$	$e_{M,\max}$ [dB]	$e_{\phi,\max}$ [deg]
G1	Direct MLLM	**0.0211**	**0.1601**	**0.8389**
	Two-stage LLM	0.0279	0.1972	1.0454
	MATLAB-N	0.0721	0.3558	4.1116
	MATLAB-S	0.1308	0.6220	6.7626
G2	Direct MLLM	0.0458	0.3887	0.4646
	Two-stage LLM	**0.0228**	**0.1943**	**0.3208**
	MATLAB-N	0.0286	0.2396	0.6067
	MATLAB-S	0.0293	0.2345	0.6182
G3	Direct MLLM	0.0256	0.2197	**0.4430**
	Two-stage LLM	**0.0231**	**0.1947**	0.5908
	MATLAB-N	0.2330	1.1741	13.2033
	MATLAB-S	0.5365	9.3179	43.9813
G4	Direct MLLM	0.0517	0.3637	**1.6836**
	Two-stage LLM	**0.0475**	**0.1909**	2.6275
	MATLAB-N	0.2771	0.2810	15.9808
	MATLAB-S	0.2484	3.0450	11.4745
G5	Direct MLLM	0.0587	0.4701	1.5679
	Two-stage LLM	**0.0119**	**0.1022**	**0.2698**
	MATLAB-N	0.0412	0.2385	2.1389
	MATLAB-S	0.0174	0.1334	0.5712
G6	Direct MLLM	0.0802	0.6616	3.7823
	Two-stage LLM	**0.0313**	**0.2181**	1.5255
	MATLAB-N	0.0385	0.2441	1.7348
	MATLAB-S	0.0341	0.2449	**1.4548**

also generates points and performs numerical optimization internally in some cases.

4.2 Transfer-Function Structure Estimation

Table 6 shows the number of order matches for each method and the RHP-zero detection results for G5. The LLM showed a high match rate for the order of the transfer function. Direct MLLM selected models with underestimated order for G6 in some cases. Two-stage LLM, in contrast, selected model with overestimated order for G2.

The Code Interpreter logs showed that candidate models were generated by taking into account response features such as the relative degree, resonances and antiresonances, time delay, and RHP zeros. This suggests that the LLM generated plausible candidates on the basis of the shape of the Bode plot, and that this contributed to the high order match rate.

In contrast, models with overestimated order relative to the true value were selected in 35 of the total 36 conditions of MATLAB-N and MATLAB-S. Figure 5 shows the result of classifying the excess roots of the models with overestimated order. Here, the relative distance in the complex plane between a pole p and a zero z is defined as $d(p,z) = |p-z|/\max(|p|,|z|)$, and a model was determined to have a close pole-zero pair when the minimum value of this distance in the model was less than 0.01. In the models with overestimated order, poles and zeros outside the evaluation frequency band and close pole-zero pairs were frequently observed, and excess roots within the band were also observed in some models. In MATLAB, candidates are selected using the maximum complex relative error with respect to the input data points as the main criterion. This suggests

Table 6. Structure estimation results over 18 trials.

Method	Order match	G5 RHP-zero detection
Direct MLLM	16/18	3/3
Two-stage LLM	**17/18**	3/3
MATLAB-N	0/18	3/3
MATLAB-S	1/18	3/3

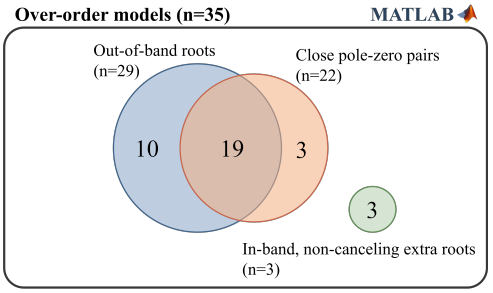

Fig. 5. Distribution of excess-root patterns in MATLAB models with overestimated order.

that these excess roots reduced the fitting error and that models with overestimated order were selected. In fact, for G3, cases were observed in which, despite the small errors at the 200 input points, close pole-zero pairs that produce narrow response variations between the sampled points were included, and the error increased in the evaluation at 10,000 points. Thus, in candidate selection based only on the numerical fit to the input data points, the structural validity such as the model order and the pole-zero configuration can be underestimated.

Table 7 shows the time-delay estimation error for G4 and the estimation error of the RHP-zero location for G5. All methods detected the presence of the time delay. Whereas the two LLM-based methods obtained estimated time delays close to the true value, MATLAB had large estimation errors. In MATLAB, although the time delay was close to the true value for the true 3rd/4th-order candidate, a candidate with overestimated order was ultimately selected. In higher-order models, the additional poles and zeros allow the rational part to represent some of the phase variation caused by the time delay within a finite frequency band. This suggests that the fitting error was reduced while the estimation error of the time delay increased. In addition, all methods detected the RHP zero for G5, and the nonminimum-phase structure itself could be estimated by both the LLM and MATLAB.

4.3 Computational Cost

Table 8 shows the means of the number of input tokens, the number of reasoning tokens, the number of output tokens, the processing time, and the API fee of each

Table 7. Estimation results for time delay and RHP zero.

Method	Mean delay error (G4) [s]	Mean RHP-zero error (G5) [rad/s]
Direct MLLM	**0.000018**	0.000465
Two-stage LLM	0.000031	**0.000346**
MATLAB-N	0.003676	0.000445
MATLAB-S	0.001880	0.000655

Table 8. Mean computational costs over 18 trials.

Method	Input tokens	Reason. tokens	Output tokens†	Time [s]	Cost [USD]
Direct MLLM	7,424	3,358	3,430	84.92	0.2600
Two-stage LLM					
Stage 1	13,012	2,978	7,188	107.29	0.4007
Stage 2	10,896	4,031	4,156	122.19	0.2991
Total	23,909	7,009	11,344	229.48	0.6999
MATLAB-N	13,012	2,978	7,188	186.95	0.4007
MATLAB-S	13,012	2,978	7,188	236.86	0.4007

† Output tokens include reasoning tokens; both are billed at the output rate. API costs were estimated using the pricing checked on August 15, 2026: USD 5 per million input tokens, USD 30 per million output tokens, and USD 0.12 per 4-GB Code Interpreter session. These values do not represent the actual billed amounts. The API fees and token counts for MATLAB-N and MATLAB-S include only the common Stage 1 components and do not include costs such as MATLAB licenses.

Table 9. Error propagation results over 18 trials.

Method	Mean κ Complex / Gain / Phase	Trials with $\kappa < 1$ Complex / Gain / Phase
Two-stage LLM	**0.732 / 0.757 / 0.791**	**14/18** / 13/18 / **14/18**
MATLAB-N	1.797 / 0.930 / 2.034	**14/18** / **14/18** / 9/18
MATLAB-S	0.908 / 0.906 / 1.038	10/18 / 12/18 / 9/18

method. Because Two-stage LLM requires two API calls, the fee and the processing time increased compared with Direct MLLM, and both the mean API fee and the mean processing time became approximately 2.7 times as large. Thus, applying the two-stage method to all inputs cannot necessarily be said to be advantageous in terms of cost-effectiveness.

4.4 Error Propagation

Table 9 shows how the point-extraction error changed as a result of the subsequent transfer-function estimation. For Two-stage LLM, the mean error amplification factor κ was less than 1 for all of the complex response, the gain, and the phase, and the trials in which $\kappa < 1$ also accounted for the majority. In contrast, for MATLAB the results depend on the metric and the estimation method.

This indicates that, for Two-stage LLM, the extraction error contained in the extracted points is smoothed by fitting a finite-order transfer function. Although the error after estimation is relatively large near the peak frequencies, the maximum error tended to be reduced.

5. CONCLUSION

In this study, we investigated a two-stage method that extracts frequency-response data points from a Bode plot image as an explicit intermediate representation and estimates a transfer function from the extracted points. The experiments showed that a high accuracy was obtained for complex-structured systems with closely spaced poles or zeros, for nonminimum-phase systems with an RHP zero, and for high-order systems, and the Two-stage estimation showed greater improvement in accuracy than the direct estimation. The transfer-function orders estimated by the LLM often matched the true orders, whereas the transfer-function orders estimated by MATLAB tended to be higher than the true orders. This indicates that, when selecting a candidate model, it is necessary to consider not only the numerical fit but also the structural validity of the model order and the pole-zero configuration. In addition, whereas the maximum error after the subsequent estimation was reduced compared with that at the point extraction in many trials, failures of the point extraction itself were also observed.

In the future, we will investigate comparative experiments that separate the effect of the self-evaluation alone, robustness evaluation including point-extraction failures, and methods and candidate selection that take into account the estimation accuracy, the computational cost, and the structural validity. Furthermore, we will investigate the extension to time-response graphs such as step responses, and the estimation of a nominal model and the uncertainty from responses that include uncertainty.

APPENDIX

The transfer functions of the six test systems are given below.

$$G_1(s) = \frac{22.3570(s+0.1995)(s^2+10.0476s+630.9591)}{(s+6.3096)(s+100)(s^2+0.2s+1)}$$

$$G_2(s) = \frac{707.9907(s+12.5893)(s+15.8489)}{(s+1.2589)(s+1.5849)(s+1.9953)} \times (s+79.4328)(s+100)$$

$$G_3(s) = \frac{281.8698(s^2+5.5735s+15.8492)}{(s^2+0.8019s+0.2512)(s^2+75.7148s+3981.0674)}$$

$$G_4(s) = G_1(s)e^{-0.01s}$$

$$G_5(s) = \frac{-22.3570(s-0.1995)(s^2+10.0476s+630.9591)}{(s+6.3096)(s+100)(s^2+0.2s+1)}$$

$$G_6(s) = \frac{3574.7180(s+1.2589)(s+50.1187) \times (s^2+0.0798s+0.03980)(s^2+5.0238s+630.9591)}{(s+0.1000)(s+0.5012)(s+2.5119)(s+6.3096) \times (s^2+5.0357s+158.4905)(s^2+20s+10000)}$$

REFERENCES

[1] Tianyi Bai et al., "A Survey of Multimodal Large Language Model from A Data-centric Perspective", arXiv:2405.16640, 2024.

[2] Jinsong Li et al., "Visual Self-Refine: A Pixel-Guided Paradigm for Accurate Chart Parsing", International Conference on Learning Representations (ICLR), arXiv:2602.16455, 2026.

[3] Xuanle Zhao et al., "ChartCoder: Advancing Multimodal Large Language Model for Chart-to-Code Generation", Proc. 63rd Annu. Meeting Assoc. Comput. Linguistics (ACL), pp.7333-7348, 2025.

[4] James K. Roberge, "Operational Amplifiers: Theory and Practice", Massachusetts Institute of Technology, LibreTexts, Sec. 3.4, "Frequency Response", compiled May 13, 2026.

[5] The MathWorks Inc., "MATLAB", Version R2026a, Natick, MA, USA, 2026.

[6] The MathWorks Inc., "tfest: Estimate Transfer Function Model", System Identification Toolbox Doc., Natick, MA, USA, accessed May 29, 2026.

[7] Ahmet Arda Ozdemir and Suat Gumussoy, "Transfer Function Estimation in System Identification Toolbox via Vector Fitting," IFAC-PapersOnLine, vol. 50, no. 1, pp. 6232-6237, 2017.

[8] OpenAI, "Code Interpreter," OpenAI API Documentation, accessed Aug. 27, 2026.