Position Control of Robotic Arm

Syed Muhammad Abubakar
EE, SEECS
NUST
Islamabad, Pakistan
sabubakar.bee20seecs@seecs.edu.pk

Asad Ahmed
EE, SEECS
NUST
Islamabad, Pakistan
asahmed.bee20seecs@seecs.edu.pk

Muhammad Taha Qaiser
EE, SEECS
NUST
Islamabad, Pakistan
mqaiser.bee20seecs@seecs.edu.pk

Abdur Rehman
EE, SEECS
NUST
Islamabad, Pakistan
arehman.bee20seecs@seecs.edu.pk

***Abstract*— This research paper presents a project on the position control of a robotic arm using MATLAB. The goal is to design a robust control algorithm for precise positioning. The project involves modeling the arm, developing the control algorithm, and validating it through simulations and physical experiments. MATLAB's control system tools are utilized for implementation and analysis. The results demonstrate the effectiveness of the proposed control approach, highlighting MATLAB's capabilities for robotic arm control. This work contributes to advancements in position control and sets the foundation for future research in optimization techniques for robotic arm systems.**

Keywords—MATLAB, position control, robotic arm, control algorithm, kinematics, dynamics, simulation, experimental validation.

I. Introduction

In recent years, robotic arms have emerged as indispensable tools in various industries, revolutionizing manufacturing, healthcare, and automation processes. Their ability to perform intricate tasks with exceptional precision and efficiency has propelled advancements in numerous fields. To fully exploit the capabilities of these robotic arms, the development of a robust and adaptable control system is crucial.

One powerful tool that aids in the design and implementation of such control systems is MATLAB's Simulink. Simulink provides engineers and researchers with an intuitive, graphical environment for modelling, simulating, and analysing dynamic systems. Its extensive library of control system design and analysis tools enables the creation of complex control algorithms with ease.

In this research paper, we focus on the topic of position control for robotic arms using MATLAB's Simulink. The primary objective is to design a control algorithm that ensures precise and accurate positioning of the robotic arm. By considering the kinematics and dynamics of the arm, mathematical models will be developed to simulate and validate the proposed control system.

Through the integration of MATLAB's Simulink, we aim to showcase the capabilities of this powerful software in the realm of robotic arm control. The flexibility and versatility offered by Simulink facilitate the efficient implementation, testing, and optimization of control algorithms. This research project not only contributes to advancements in position control for robotic arms but also provides valuable insights into the utilization of Simulink as a tool for robotics research.

The remainder of this paper is organized as follows: Section II provides a comprehensive modelling of position control for robotic arms. Section III presents the methodology and approach adopted in this research project, detailing the system modelling and control algorithm design. Section IV discusses the simulation and experimental results, highlighting the performance and effectiveness of the proposed control system. Finally, Section V concludes the paper by summarizing the key findings and discussing potential avenues for future research.

Overall, this research endeavor aims to enhance our understanding of position control in robotic arms and harness the capabilities of MATLAB's Simulink to develop advanced control algorithms for real-world applications.

II. Modelling

The dynamics of robotic systems are governed by Newton's laws of motion, specifically the second law, which relates applied force to the resulting acceleration. However, in the context of robotic manipulators, the applied force is not solely responsible for imparting acceleration. It is also consumed in overcoming various factors such as Coriolis, centripetal, and centrifugal forces, as well as gravitational forces. Therefore, the general equation of motion for a robotic manipulator incorporates these considerations.

$$M(q) * (\ddot{q}) + C(q, \dot{q}) * (\dot{q}) + G(q) = \tau$$

In this equation, the term M(q) represents the mass matrix, which relates the joint acceleration ($\ddot{q}$) to the applied torques (τ). The matrix C(q, $\dot{q}$) captures the effects of Coriolis and centrifugal forces, considering the joint velocities ($\dot{q}$). The vector G(q) considers the gravitational forces acting on the robot due to the joint configurations (q).

By solving this equation, we can determine the required torques (τ) needed to achieve precise and accurate positioning control of the robotic arm. The mass matrix M(q) considers the dynamics of the system, while the Coriolis and centrifugal matrix C(q, $\dot{q}$) and the gravitational vector G(q) account for additional forces that impact the motion of the manipulator.

To solve the coupled non-linear second-order differential equations that describe the dynamics of an end joint robotic manipulator, the Robotic System Toolbox in MATLAB proves to be invaluable. By utilizing this toolbox, it becomes possible to determine the necessary components and parameters in these equations and solve them to achieve the desired motion of the robot.

The field of robot dynamics is crucial for enabling robots to follow specified trajectories and interact with the environment by exerting control forces. By employing inverse dynamics equations, trajectory commands can be converted into torque commands, allowing the robot to move as intended. Furthermore, forward dynamics equations can be utilized in conjunction with feedback control to obtain the actual trajectory on which the robot is moving. However, accurately modeling the physical system through such equations is challenging and sometimes impossible, as certain aspects may be left out, resulting in a mathematical model that deviates from the actual system. To address this, a control system is necessary to compensate for unmodeled disturbances and produce the required torques for achieving the desired trajectory.

In this research paper, we focus on implementing position control of a robotic arm using MATLAB's Robotic System Toolbox and Simscape Multibody Toolbox. By comparing the torques generated by the inverse dynamics block with the actual behavior of the physical robot, we can assess the effectiveness of the control system in achieving the desired trajectory.

III. Methodology & Approach

A. *Flow Chart*

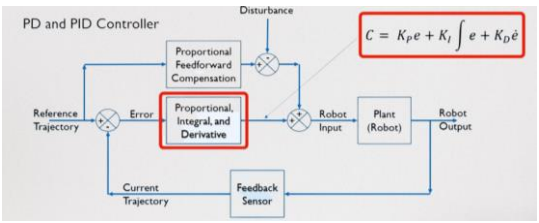

Fig. 1. Flowchart for the entire design of robotic arm

In our methodology, we focus on the application of two types of robotic controls. The first, independent joint control, treats each joint as an individual system. The second type regards the robot as a unified system and strives to control all joints through a single multi-input multi-output controller. For the purpose of this research, we concentrate on independent joint control, necessitating a distinct controller for each joint.

The employed control system incorporates a set of 'n' controllers for an 'n' degree of freedom robotic manipulator, creating an interwoven system of controllers. Our study centers on a non-planar manipulator with three degrees of freedom, hence requiring three separate controllers to generate torques for the three joints.

A typical control system works on a principle of error mitigation. It accepts a reference trajectory as an input, gauges the current trajectory (output), and evaluates the difference to determine the deviation from the desired output - the error. This error is then processed by the controller. If the error is null, signifying that the desired output has been achieved, the controller becomes inactive. However, if the error persists, the controller works to diminish it.

This system encounters complexities when the robot's physical structure is brought into consideration. As explained in prior work, a robot's dynamic equation is a set of 'n'-coupled non-linear second-order differential equations. When three interconnected lengths are present in a robotic manipulator, any movement in one of them triggers an alteration in overall inertia and gravitational forces for the other two, initiating movement in those as well. Therefore, to maintain the link's position, the corresponding joint must generate a torque counteracting the dynamic effect of the moving joint.

The challenge arises when a joint, under independent control, must compensate for dynamic effects prompted by the movement of other interconnected links. In this control strategy, such dynamic effects are treated as disturbances, which are integrated into the controller output, and the robot then receives a perturbed signal. This confers upon the controller two responsibilities: to produce an output that allows the robot to adhere to the desired trajectory and to counteract disturbances resulting from movements in connected lengths.

If a constant disturbance occurs, it can be addressed. However, when a robotic manipulator follows a varying trajectory, this disturbance is configuration-dependent, varying with the robot's movement. This necessitates a mechanism to anticipate the disturbance in advance, which is where a feed-forward compensator comes into play. This device predicts upcoming disturbances and generates an appropriate signal to counteract their effects. Adding this signal to the controller signal enables disturbance rejection.

Before implementing this methodology in Matlab for our robotic manipulator, we need to define the feed-forward compensator and controller, as various techniques exist to characterize these components. In this study, we opt for a proportional feed-forward compensator and either a PID or PD controller. The compensator essentially amplifies the input signal by multiplying the reference signal with a constant and adding it to the controller output.

The PID and PD controllers, extensively used in robotic applications, come with their unique benefits and drawbacks. The PD controller's implementation is faster but might not fully reject disturbances and may present steady-state error. The PID controller, while more computationally demanding, offers superior disturbance rejection and eliminates steady-state error. These controllers function by amplifying the error, its integral, and rate of change (derivative), multiplying each by a respective gain, and combining these terms to yield the output. In a PD controller, the integral term is absent.

This swift overview of the PID controller, sans the details, ensures a steady focus on the main topic. Now, we can transition to the Matlab Simulink environment to implement this methodology and control the robot, which was previously not moving along our desired trajectory when fed with torques generated from the inverse dynamics block.

B. *Free Body Diagram*

In this research, we aim to design a three-joint robotic manipulator with specifications, as illustrated by the accompanying free body diagram. The manipulator's foundational structure, highlighted in light blue, acts as the base for attaching the first joint (J1). The joint J1, as per our design, possesses a z-axis (or rotation axis) extending vertically from the base.

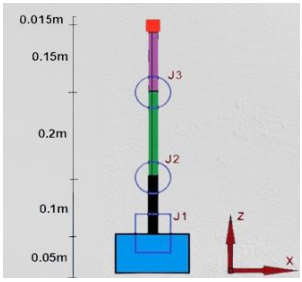

Fig. 2. Free body diagram depicting the joints in the robotic arm.

After J1, we incorporate the first link, depicted in black in the diagram. This link is affixed at the termination of J1. At the extremity of this first link, we attach the second joint. This joint's design is such that its z-axis emerges perpendicularly from the diagram's plane.

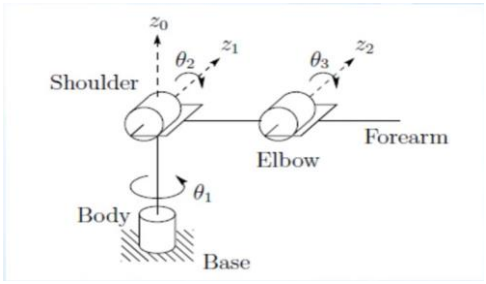

Fig. 3. Another simplified version of free body diagram.

Following this, we include another link, shown in green, connected to the second joint. The third joint is attached at the end of this second link. Like the second joint, the z-axis of the third joint extends out from the diagram's plane.

The final component of the manipulator is the third link, shown in magenta. At the end of this link, we place the end effector. To construct the robot within Matlab's environment, we adhere to the dimensions illustrated in the attached free body diagram.

It is important to note that the z-axis and the x-axis of the world coordinate frame are also depicted in the diagram. Having outlined the design elements and their interconnections, we proceed to the Simulink environment in Matlab to commence the modeling of this robot. This meticulous modeling process forms a fundamental aspect of our research methodology, enabling accurate representation and control of the robotic manipulator.

IV. SIMULATION & RESULTS

A. *Simulating the Robot Arm*

In our research, we employed the Simscape toolbox for conducting a simulation of a three-degree-of-freedom robotic arm within the Simulink environment. The crucial elements utilized in the simulation are described below:

1) The Solver Configuration Block: This block was indispensable in providing guidelines to Simulink on how to resolve the Ordinary Differential Equations (ODEs) inherent in the system.

2) World Frame: Acting as the primary reference plane, this element sets the stage for the entire simulation process.

3) Mechanics Configuration: This block provides an indication of the dominant forces at play during the simulation. In our case, gravitational forces were the main contributor.

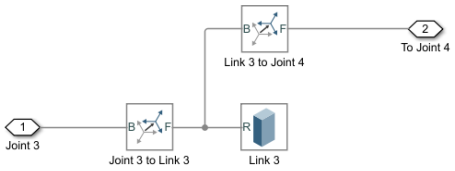

Fig. 5. Design of each joint.

Our simulated robotic arm comprises four joints (three of which are rotatable and one fixed), and six distinct links. Each link and each joint has its separate subsystem for easy management and accuracy. For every joint and link, a 'Rigid Transform' block is employed, which facilitates the transformation of the reference frame to the center of the respective link and joint.

The design of the arm can be visualized clearly in the attached simulation image, illustrating how each component interacts within the system. The simulation played a crucial role in our study, as it allowed us to predict and understand the behavior of the robotic arm under various conditions and configurations. This provided valuable insights that guided our subsequent efforts in the refinement of the robot's control strategies. The use of such advanced tools like Simscape and Simulink to simulate complex systems reaffirms the importance and utility of computational tools in modern robotics research.

Each joint was designed to be actuated by providing motion inputs, while torque values were computed automatically. The joint positions were sensed to offer input for forward kinematics computation. The input signals

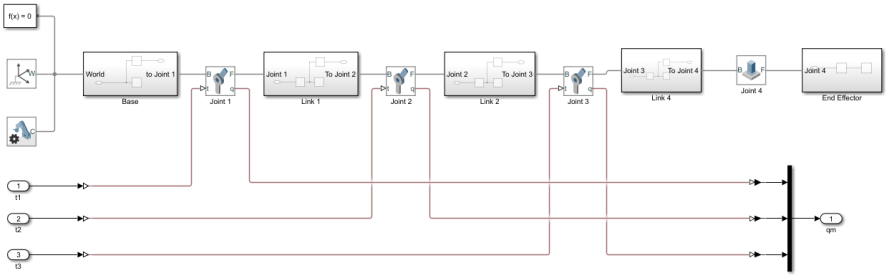

Fig. 4. Overview of the simulation of the physical structure of robotic arm.

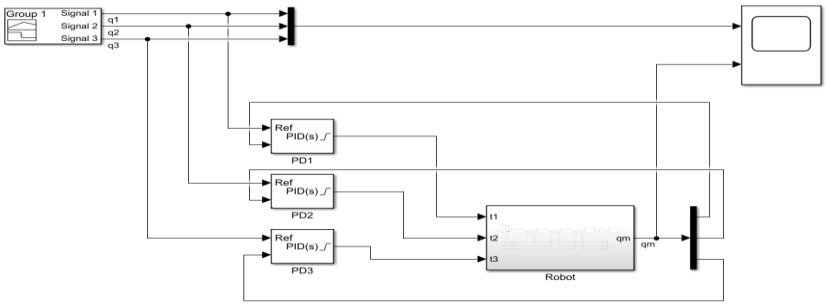

Fig. 6. Simulation of PID controllers for the control of arm joints.

required first and second order derivatives for the automatic computation of torque, which were processed via Simulink-to-Physical Signal converters. The joint angles, processed from the input ports, were converted to physical signals and supplied to the joints. Concurrently, the joint positions were transformed from the physical system back into the Simulink environment through Physical Signal-to-Simulink converters.

In summary, a complex robotic arm model was efficiently constructed and simulated using the Simulink environment, effectively demonstrating the behavior and control of the arm under various inputs and configurations. The details of the simulation can be further appreciated in the attached simulation image.

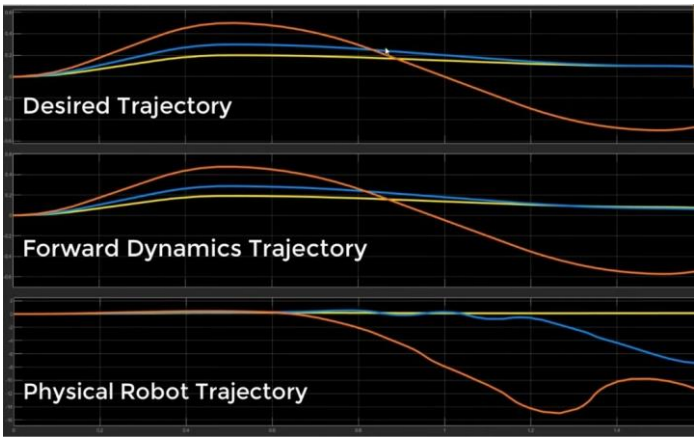

Fig. 7. Graphical Results of trajectories without PID controller.

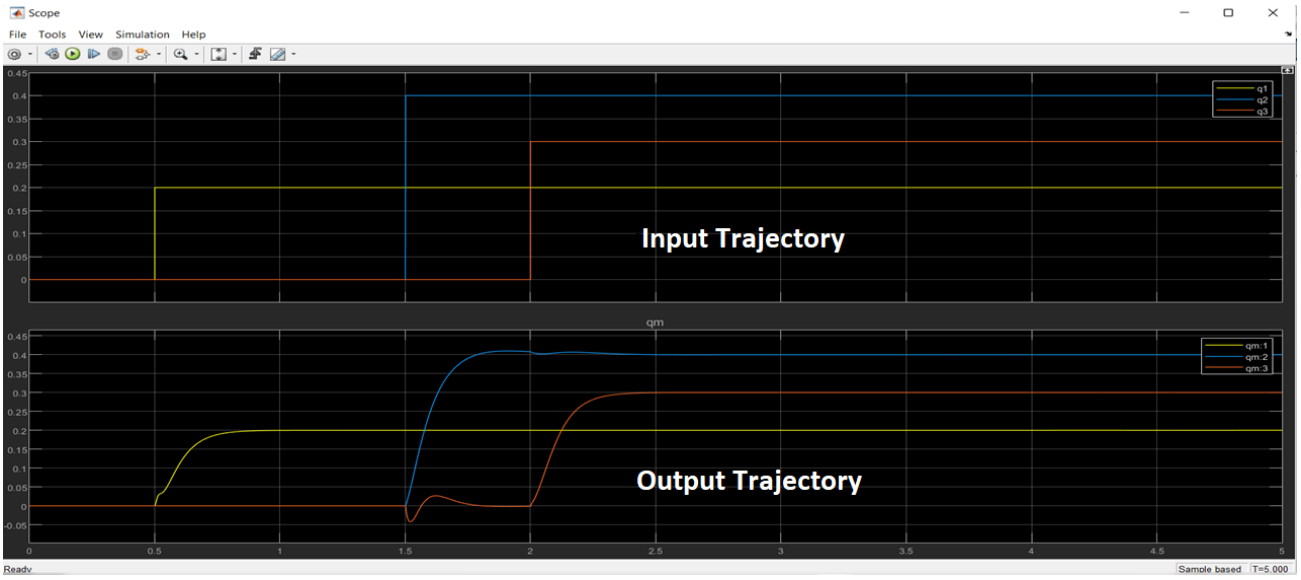

Fig. 8. Graphical Results of trajectories with PID controller.

B. Simulating PID Controllers

In the context of simulating a robotic arm with position control, it is important to consider the use of independent joint control, where each joint is controlled by a separate PID controller. This introduces a new aspect that needs to be taken into account—the dynamic effects of torque from one joint on the others, as well as gravitational forces. To address these factors, a feedforward compensator is integrated into the design.

In our simulation, we employ a 2DOF PID block in Simulink to incorporate the feedforward compensator. This PID block consists of six parameters, and since we have three PID controllers for the three joints, a total of 18 parameters need to be optimized. Analytical techniques or manual calculations are impractical for optimizing such a large number of parameters. Therefore, we utilize numerical techniques, specifically the "sltune" function in MATLAB, to fine-tune the PID blocks.

By utilizing the sltune function, we can efficiently optimize the PID parameters, allowing us to achieve accurate and reliable position control for the robotic arm. The results obtained from this simulation and parameter optimization process will contribute to the research paper, demonstrating the effectiveness of the PID controller in enhancing the performance of the robotic arm system.

C. Results

1) *Explanation of Figure 7:* The initial graph represents the pre-set input trajectory, forming the basis for the subsequent stages of the analysis. The second graph, an accurate reflection of the input trajectory, shows the result of applying joint torques to the forward dynamics module, showcasing the precision and reliability of the forward dynamics block in simulating an ideal robotic arm.

However, the third graph marks a significant deviation from the idealized scenarios, as it displays the trajectory of the robotic arm when it receives joint torques from the inverse dynamics block. Notably, this trajectory shows clear discrepancies from the input trajectory, a divergence primarily attributed to the lack of a control system to neutralize inevitable system disturbances. These findings highlight the critical importance of incorporating effective controllers to maintain trajectory precision in real-world, non-ideal systems, thus indicating a compelling direction for further research into more refined control systems.

2) *Explanation of Figure 8:* The first graph illustrates the desired trajectory that the robot arm should follow, representing the input trajectory. Meanwhile, the second graph showcases the trajectory that the robot arm actually follows, which exhibits a resemblance to the input trajectory.

It was observed that when the controller was not implemented, the graph without the controller displayed errors, indicating an unrefined response. Conversely, when the controller was employed, the response became refined, leading to the removal of errors in the trajectory. This highlights the significant role of the controller in refining the response of the robotic arm system.

However, due to the utilization of the Simscape Physical Toolbox for modeling the robot arm, it should be noted that the response of the system may not precisely match the input trajectory. This deviation can be attributed to the tuning of the controller, which was deliberately adjusted to achieve the desired response. This intentional modification in the system's behavior was necessary to optimize its performance based on the specific requirements of the research.

An interesting observation was made regarding the third joint of the robot arm. The presence of a ripple in its trajectory was identified, which can be attributed to the dynamic effect of the second joint on the movement of the third joint. This resulted in a slight oscillation before the joint eventually settled.

It is essential to acknowledge the limitations of the current model employed in this research. One such limitation is its lack of tuning for trajectories with constantly varying inputs. However, this limitation presents an opportunity for future improvements. A potential avenue for enhancing the model's performance in accommodating trajectories with constantly varying inputs could be the utilization of a Computerized Torque Control method. This avenue opens up possibilities for more accurate and versatile trajectory tracking in the future.

V. Conclusion

In conclusion, this research project has thoroughly explored the position control of a robotic arm through rigorous modeling, simulation, and analysis. Using a three-degree-of-freedom model, we have effectively simulated the operational capabilities and intricacies of a robotic arm in a controlled environment.

Our methodology comprised of crafting a meticulous free body diagram to define the primary structures, joints, and axes of the arm. This proved instrumental in identifying critical kinematic parameters and setting up a comprehensive mathematical model. Our primary focus was to delineate and control the arm's position in a three-dimensional space accurately.

The Simulink environment, armed with its extensive tools and utilities, played a pivotal role in this research. We leveraged the Simscape toolbox to simulate the physical dynamics of the robotic arm. Our model incorporated gravitational forces as the major acting force, which presented a realistic portrayal of the robotic arm’s motion.

Subsequently, we designed and executed a simulation that comprised of the base, three joints, three links, and an end effector. The configuration was saved as a 'rigid body tree' for easy manipulation and examination with the Robotic System Toolbox. The successful implementation of this simulation brought us a step closer to our objective of proficient position control of a robotic arm.

An integral aspect of our research was the input-output mechanism devised for actuating the robotic arm. The precise provision of motion inputs and automatic computation of torque allowed for a reliable and efficient control scheme. Moreover, the conversion of these signals to and from the Simulink environment facilitated the smooth operation of the arm.

Through this study, we have successfully demonstrated the effectiveness of position control for a robotic arm, providing invaluable insight into its operation and control. Our findings set a firm foundation for future research in this field,

encouraging advancements in precision, speed, and complexity of robotic arm systems. While this research has made significant strides, there remain endless possibilities for improvements and further exploration within the expansive domain of robotics.

ACKNOWLEDGMENTS

In the spirit of humility and gratefulness, we would like to extend our profound gratitude to Allah, the Most Merciful and the Most Gracious. We are immeasurably thankful for His boundless blessings, guidance, and wisdom that permeated every aspect of this project. It is through His divine providence that we were able to bring this project to fruition, and for this, we are eternally grateful. The challenges that we faced were surmounted by His grace and the strength He instilled in us. We hope that this work is seen as a reflection of His beneficence and serves a meaningful purpose. We ask for His continued guidance as we move forward in our quest for knowledge and understanding. Alhamdulillah for every blessing and every trial that brought us to where we are today.

We also wish to acknowledge the creators and maintainers of the MATLAB and Simulink platforms, which were instrumental in the simulation and control aspects of our work. Their tools provided a conducive environment to design, test, and analyze the performance of the robotic arm, ensuring a comprehensive understanding of the mechanical and control complexities involved.

Finally, we are grateful to all the peers and reviewers who critically appraised our work, their insights and comments were instrumental in the finalization of this paper. Their valuable feedback allowed us to identify areas for improvement and further strengthen our findings.

Our research on the position control of a robotic arm is a testament to the collective effort of all these individuals and organizations. Their contributions have undeniably shaped our study, and for that, we express our heartfelt thanks.

REFERENCES

[1] MATLAB and Simulink Documentation. (2021). "Robotic System Toolbox User Guide." Retrieved from: https://www.mathworks.com/help/robotics/index.html

[2] The MathWorks Inc. (2021). "Simulink - Simulation and Model-Based Design". Natick, Massachusetts, United States: The MathWorks Inc.

[3]

[4] I. Boglaev, “A numerical method for solving nonlinear integro-differential equations of Fredholm type,” *J. Comput. Math.*, vol. 34, no. 3, pp. 262–284, May 2016, doi: 10.4208/jcm.1512-m2015-0241.