ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR03

HEXAPOD CONTROL SYSTEM DEVELOPMENT TOWARDS ARBITRARY TRAJECTORIES SCANS AT SIRIUS/LNLS

A. Y. Horita *, F. A. Del Nero, M. A. L. Moraes, G. N. Kontogiorgos, LNLS, Campinas, Brazil
G. G. Silva, UNICAMP, Campinas, Brazil

Modern 4th generation synchrotron facilities demand high precision and dynamic manipulation systems capable of fine position control, aiming to improve the resolution and perfomance of their experiments. In this context, hexapods are widely used to obtain a flexible and accurate 6 Degrees of Freedom (DoF) positioning system, since it is based on Parallel Kinematic Mechanisms (PKM). Aiming the customization and governability of this type of motion control system, a software application was entirely modeled and implemented at Sirius. A Bestec hexapod was used and the control logic was embedded into an Omron Delta Tau Power Brick towards the standardization of Sirius control solutions with features which completely fill the beamline scan needs, e.g., tracing arbitrary trajectories. Newton-Raphson numerical method was applied to implement the PKM. Besides, the kinematics was implemented in C language, targeting a better runtime performance when comparing to script languages. This paper describes the design and implementation methods used in this control application development and presents its resulting performance.

INTRODUCTION

Sirius is the 4th generation synchrotron light source being commissioned by the Brazilian Synchrotron Light Laboratory (LNLS), in Brazil [1]. Its low emittance (0.25 nm rad) makes it one of the world's brightest light sources of its kind [2]. In this sense, high precision control systems are required towards better beam quality and stability.

In Sirius IPE Beamline, a Bestec P468 Mirror Unit [3] was chosen as the motion system of a toroidal mirror. This unit contains a hexapod in its structure, which is illustrated in Fig. 1. Hexapods are widely used due to the pararell control capacity of its 6 degree of freedom (DoF), characterized in Pararell Kinematics Machines (PKM) [4].

Together with this unit, a Bestec P494 Motion Control System was also acquired, which embeds a Mint Linux ditribution as Operating System (OS), containing a set of proprietary control software which implements the user interface, configuration files, system kinematics and the hexapod system interface.

Although the P494 control system meets the beamline expected stability and motion control accuracy, a customized system developed in-house was desired towards Sirius's systems standardization and independence when implementing new features, such as arbitrary trajectories scans. Under these circumstances, we decided to implement the control logics using an Omron Delta Tau Power Brick LV controller [5].

* augusto.horita@lnls.br

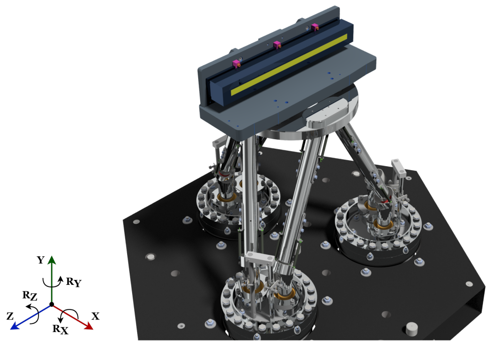


Figure 1: Toroidal mirror supported by hexapod in the Bestec P468 Mirror Unit.

An important portion of the motion control logics consists of its kinematics, which is splitted into two algorithms.The inverse kinematics converts system positions to individual motors' positions and the forward kinematics calculates the system feedback positions based on encoders [6]. Towards a more productive and robust implementation process, the hexapod kinematics was first mathematically modeled using a Jupyter Notebook [7], which provided flexibility using software structures, e.g. mathematical libraries, and optimized the simulation and bugfixes process [8]. Due to the fact that the kinematics of a hexapod is non-linear [6], our model was based on Newton Rhapson for root-finding, as it is widely used and has a fast convergence rate [9].

Based on the validated Jupyter Notebook model, it was possible to implement the control logics in the Delta Tau Power Brick LV system. Its performance was then compared to Bestec's control sytem for validation. Also, an example of arbitrary trajectory was implemented for functionality validation purposes.

The remaining sections of this paper presents in more details the used methods for this control system development and its validation.

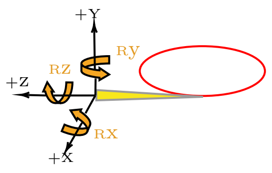
MOTION CONTROL DEVELOPMENT

In this section, we present the motion control system development, focusing on the kinematics modeling and implementation.

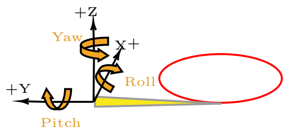
Kinematics Model

In the presented application, the toroidal mirror is placed inside a vacuum chamber. Towards a more robust and easy for maintenance architecture, Bestec's P468 Mirror Unit was designed placing the actuators outside the vacuum environment. In this context, this Mirror Unit hexapod design slightly differs from the usual hexapod configuration found in other applications and in literature [6], as illustrated in Fig. 1. In this design, the six manipulator's struts have constant length, and the platform movements are caused by the vertical displacement of six linear actuators placed outside the chamber, and connected to the struts by a spherical joint. As a consequence, the presented kinematics model also differs from models found in the literature.

For standardization purpuses, the kinematics model and the mathematical formulation presented is this work are based on the Sirius coordinate system, in which the *Z*-axis points to beamlight direction, *X*-axis outward from the storage ring and *Y*-axis is pointed upwards, as shown in Fig. 2a. The Bestec motion control sytem is based on a different coordinate system, as illustrated in Fig. 2b. This difference is compensated during the data analysis by comparing the correspondent DoF of each control system. Aiming the simplification of methods description, the remaining of this paper refers to the Sirius coordinate system.



(a) Sirius



(b) Bestec

Figure 2: Coordinate system conventions

Two auxiliary frames are used to model the kinematics transformations: the platform's coordinate system $O_p(u,v,w)$ (at platform centroid) and the granite plate coordinate system $O_g(x,y,z)$. In addition, a set of vectors is also necessary, as illustrated in Fig. 3. Let $\vec{b}_i$ (i ∈ 1, …, 6) be the vectors from O_g to the i^{th} granite joint. Similarly, let $\vec{a}_i$ (i ∈ 1, …, 6) be the vectors from O_p to the i^{th} platform joint, written with respect to platform's coordinate system. Moreover, let $\vec{l}_i$ be the vector representing the i^{th} hexapod strut, with ($\|\vec{l}_i\|$ being constant. The $\vec{s}_i$ links the i^{th} granite joint to the i^{th} platform joint, and $\vec{c}_i$ the vertical vector representing the position of the i^{th} linear actuator, in reference to the granite plate. Finally, let $\vec{p}$ be the position vector of the platform centroid O_p in reference to O_g and $\vec{m}$ the position vector of the end-effector *i.e.*, the mirror centre, considering the O_g coordinate system.

We also define a rotation matrix R as the product of three consecutive rotations along each coordinate axis *i.e.*, $R = R_x(\alpha)R_y(\beta)R_z(\gamma)$, considering that:

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad (1)$$

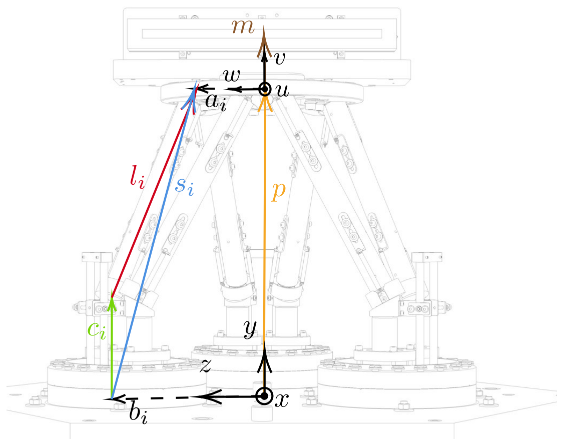


Figure 3: Coordinate systems and vectors applied to hexapod model.

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \quad (2)$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Inverse Kinematics Model Given a position $(x,y,z,\alpha,\beta,\gamma)$ of the mirror centre, we present the algorithm to find each linear actuator displacement. Based on Fig. 3, and considering that the platform and the mirror are coupled, their angular position (α,β,γ) are the same. In this sense, it is possible to find the platform's centroid O_p position, represented by vector $\vec{p}$, by subtracting the vector $\vec{m}$, of fixed module $\|\vec{m}\|$, from mirror centre position (x,y,z). Figure 3 also illustrates that vector $\vec{s}$ can be calculated as shown in Eqs. (4) and (5):

$$\vec{s}_i = \vec{p} + R\cdot\vec{a}_i - \vec{b}_i \quad (4)$$

$$\vec{s}_i = \vec{c}_i + \vec{l}_i \quad (5)$$

Considering $\vec{c}_i = (0, y_{ci}, 0)$, we can write:

$$\vec{s}_i - \vec{c}_i = (x_s, y_s - y_c, z_s) \quad (6)$$

As the hexapod struts lengths $\|\vec{l}_i\| = \|\vec{s}_i - \vec{c}_i\|$ are constant:

$$y_{ci} = y_{si} - \sqrt{\|\vec{l}_i\|^2 - z_{si}^2 - x_{si}^2} \quad (7)$$

In summary, in order to calculate the vertical displacement of the linear actuators, one may calculate the s vector using Eq. (4) and then use this result in Eq. (7).

Forward Kinematics Model In the forward kinematics algorithm, the position of the end-effector is calculated based on the given linear actuators positions, measured by absolute encoders.

From Eqs. (4) and (5), we can write:

$$c_i = \|\vec{p} + R \cdot \vec{a}_i - \vec{b}_i - \vec{s}_i + \vec{l}_i\| \tag{8}$$

Which represents a set of six nonlinear equations that composes the forward kinematic model. From Eq. (8) we obtain the position of the end-effector traversing the $\|\vec{m}\|$ distance in the direction perpendicular to the platform ($\hat{v}$ direction). A possible approach for solving these equations is to use a numerical method. In this context, the Newton-Raphson method was chosen for root-finding, as it is widely used in similar problems due to its convergence rate and accuracy performance.

The Newton-Raphson is an iterative algorithm, therefore Eq. (9) defines $f_i(x)$ as the square error between the given actuators position and the calculated position at the k^{th} iteration:

$$f_i(x) = (y_i)^2 - (y_i^{(k)})^2 = 0, \quad i = 1, \ldots, 6 \tag{9}$$

Where the x variable denotes the six input variables vector, representing the end-effector coordinates, and y_i denotes i^{th} output variable, which is the vertical displacement of the i^{th} linear actuator. Let $F(x)$ be the vector of f_i functions. Given an estimate $x^{(k)}$ for the platform centroid position at the k^{th} iteration, the linear approximation around $x^{(k)}$ can be calculated as:

$$F(x) = F(x^{(k)}) + (x - x^{(k)})F'(x^{(k)}) \tag{10}$$

Where $F'(x) = (J)_{ij} = \frac{\partial f_i}{\partial x_j}$. Hence, the Jacobian matrix is defined as:

$$J = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_6(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_6(\mathbf{x})}{\partial x_6} \end{bmatrix} \tag{11}$$

Then, the input vector at the $k^{th} + 1$ iteration is calculated as:

$$x^{(k+1)} = x^{(k)} + J^{-1}(x^{(k)})F(x^{(k)}) \tag{12}$$

The algorithm proceeds until the convergence criterion $\|F(x^{(k)})\| < \epsilon$ is met or until the maximum number of 10 iterations is reached.

Kinematics Implementation

The Delta Tau integrated development environment (IDE) supports two different ways to implement the control system kinematics. It can be done using the kinematics scripts (plc files), or implementing in C language, which is indicated when the application requires heavy mathematical calculations, due to the fact that compiled languages have better runtime performance when compared to scripted ones. In this sense, our choice was to implement the C source coded kinematics.

There are some obstacles when implementing the C kinematics in Delta Tau system that had to be overcame. The Delta Tau IDE doesn't support the usage of additional C libraries, only the ones included in the software development kit (SDK), which made it necessary to adapt and use the proprietary functions included in SDK. In addition to that, the C kinematics logics have to be implemented in a specific function already defined in the SDK, the *CfromScript* function, which is called inside the script files.

On the other hand, the SDK includes functions that were useful aiming a motion control better perfomance. The *SegMoveTime* function splits the movement trajectory into time intervals of configured milliseconds. Before each segment execution, the kinematics are re-calculated. This improves the movements accuracy during the transient state by reducing the parasitic movements in other DoFs, although it costs computation resources. If the interval is set too low, there is a risk of starvation, where the controller may not be able to do all of its move calculations in the time allotted, stopping the motion program with a run-time error. As our application targets better positioning precision, we decided to use 1 ms inteval configuration, which has shown to be enough for motion calculations during our tests.

VALIDATION TESTS

The present section outlines the implemented motion control system validation tests, which includes its comparison with Bestec's P494 Motion Control System benchmark. Furthermore, it describes a simple example of the desired arbitrary trajectory functionality implementation.

Performance Tests

Aiming the standardization of control systems for Sirius beamlines' PKMs, such as granite bases, hexapods and tripods, a motion control system was developed in in-House. For the validation of this system, comparative tests were performed against the Bestec P494 motion control system.

These tests consisted of making controlled movements in one hexapod DoF, setting the same motion range and speed in both control systems, while logging all DoF's positions to check the amplitude of parasitic movements. The movements were performed using one control platform at a time. The unit positions were logged, then acquired and analyzed afterwards. Due to the fact that P494 controller sample rate for the hexapod position is limited to 10 samples/s, we configured the Power Brick LV to acquire at the same sample rate.

For compilation purposes, this paper only presents two cases of the conducted comparative performance tests, one representing a translation DoF command and another illustrating a rotation DoF command.

Figure 4 depicts the comparison between the control systems performances when commanding the unit to move from

−1 mm to 1 mm in *Z*-axis. The *Z*-axis curve shows that the control systems reached stability at the target position with a difference of approximately 0.4 s. Although P494 control reaches the destination before, it presents significant position deviations at the transient state for all DoFs, while the Power Brick LV shows smaller control errors. The Fig. 4 curves also shows that cross-talk movement ranges were smaller for the Power Brick LV system. In *X*, *Y*, R_x and R_z axes, the noticed differences represent one order of magnitude.

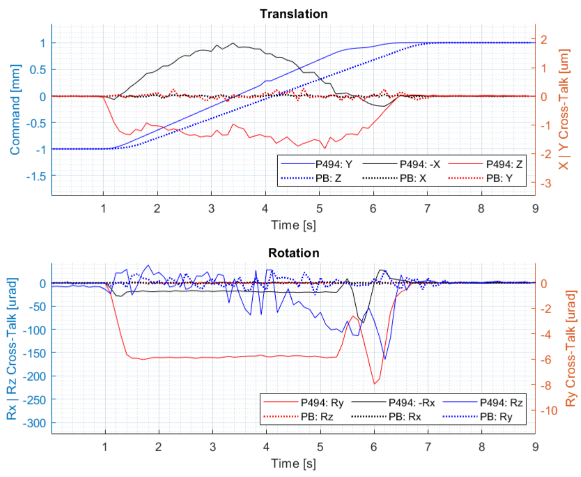


Figure 4: *Z*-axis translation trajectory and parasitic motion comparison between Delta Tau Power Brick and Bestec P494 control systems.

The command of R_z-axis moving from −3 mrad to 3 mrad was the chosen rotation DoF, which is shown in Fig. 5. The R_z-axis curve exposes a similar behavior to the previous test. The P494 system reaches the target approximately 0.3 s before, although it presents position deviations, while the Power Brick course the trajectory without position hiccups. The remaining curves illustrate that parasitic movement ranges are smaller using the Power Brick LV system for all axes. In X and Y cases, the differences represent one order of magnitude. The rotation plot also shows that P464 may present stationary error before the movement, which is noted in $P494 : -Rx$ curve.

Arbitrary Trajectory Test

Many beamline applications are not covered by point-to-point traditional s-curve trajectories. For instance, if coordinated motion is required and the hexapod should follow an external motor towards a polynomial relation, the motion profile will demand velocities' profiles other than smooth trapezoids. Or if a crystal should be rotated to vary the beam incidence angle and to perform a motion profile whose energy derivative is trapezoidal, an arbitrary trajectory is needed. If all virtual axes of an hexapod can be controlled with low cross-talk and also to perform arbitrary trajectories, then a powerful tool for fly-scans was achieved.

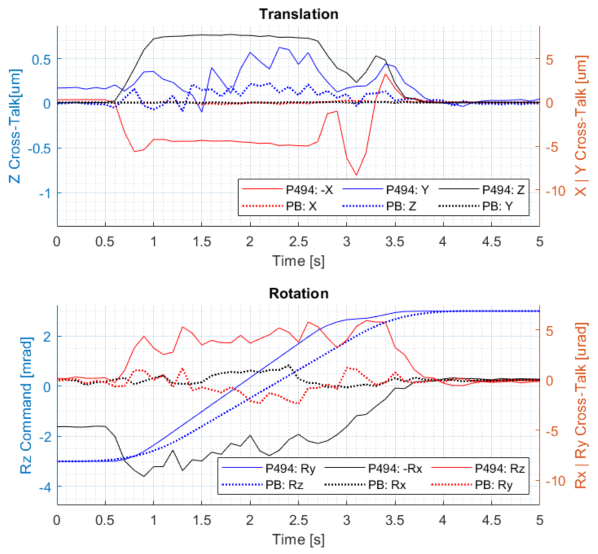


Figure 5: *Z*-axis rotation trajectory and parasitic motion comparison between Delta Tau Power Brick and Bestec P494 control systems.

In this context, the arbitrary trajectory functionality consists of performing system movements based on a previously defined sequence of DoFs positions and movements parameters, such as speed and step triggers. This may enhance the system runtime time performance and determinism, as a portion of the necessary calculations is done before the system motion execution.

When using this functionality, the trajectory parameters may be inputted into the system using different resources, e.g. they may be saved into the device memory and read during application runtime by a script, processed by a motion application. Another option is to use the EPICS interface through waveforms Process Variable (PV) to input the trajectory parameters. This paper presents a simple validation example of the second method, using a customized version of Power Brick LV and EPICS source codes implemented by Diamond Light Source (DLS) [10], which inputs the trajectory positions, time spent in each position and the speed curve mode in each step move.

For this test, the trajectory parameters were previously generated using a python script. Figure 6 illustrates the planned trajectory, which commands the hexapod to move in the 3 translations DoF.

Aiming a more realistic application, the generated trajectory also variates the speed and acceleration during the movement, which may be useful during beamline fly-scans. In this sense, a sinusoidal-exponential signal was chosen as arbitrary acceleration profile, in order to create a constantly varying velocity profile. Each trajectory edge is composed by two axes moves, one is a linear 0.001 mm increment or decrement in the range from 0 to 0.5 mm, and another is described by Eq. (13).

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOBR03

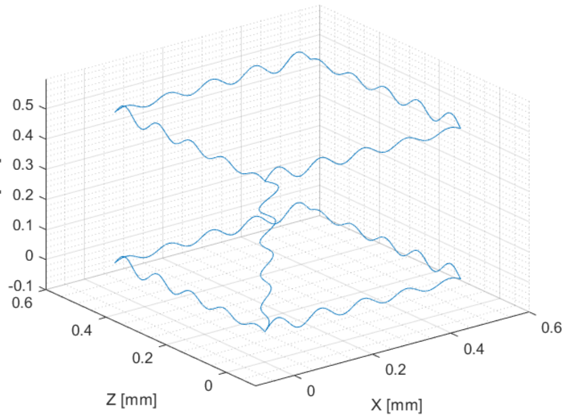


Figure 6: Generated arbitrary trajectory

$$pos_i = \frac{e^{(2*i)} * sin(i) + 2 * i}{100} \tag{13}$$

Figure 7 illustrates the generated position, speed and acceleration data considering the non-linear movement component of a single trajectory edge movement.

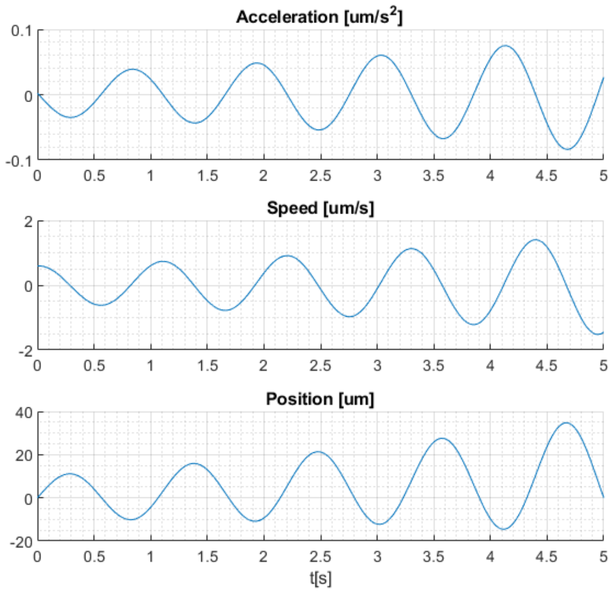


Figure 7: Generated position, speed and acceleration

Based on the generated trajectory, the DLS algorithm was used to run the application and move the hexapod motion unit. Figure 8 depicts the motion system logged position in each translation DoF during the movement, while Fig. 9 shows that the position errors of each commanded DoF during the transient state ranges from −3 to 3 µm.

The analysis of the logged position data in comparison to the planned trajectory, in addition to the low range of the computed errors, shows that the motion system follows the planned motion, which validates the arbitrary trajectory functionality implementation.

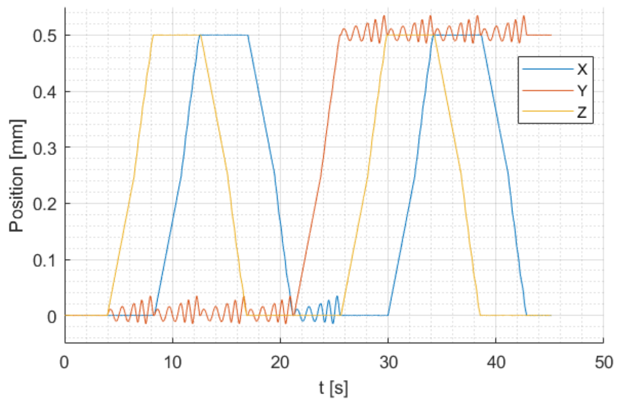


Figure 8: Time plot of arbitrary trajectory logged positions.

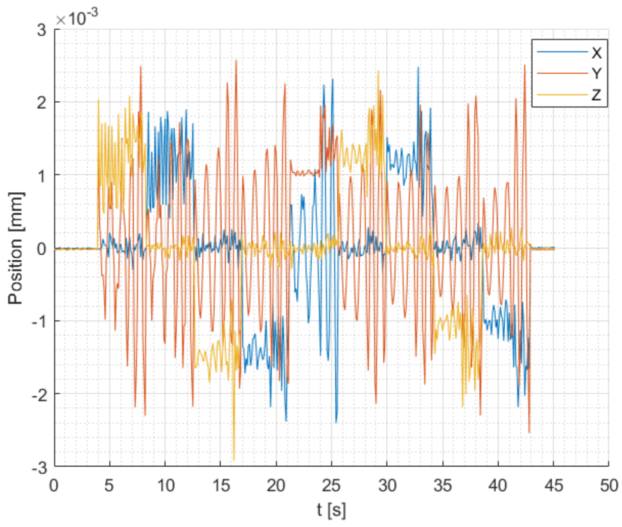


Figure 9: Time plot of transient state position errors.

CONCLUSION

A motion control system for the Bestec P468 Mirror Unit hexapod was implemented using Delta Tau Power Brick LV, a standard motion controller used in Sirius. Although the Bestec P494 system can reach its specifications at steady state and perform step-scans, it is neither suitable for fly-scans nor distributed coordinated motion, as relevant cross-talks are present during commanded movements. The performance tests demonstrate that the in-house designed system's motion control accurary at transient state overcomes the manufacturer one, specially when comparing the parasitic movements during a single DoF command, which is fundamental for fly-scans, where the detectors acquire images during transient motions. In addition, the arbitrary trajectory functionality was validated using the new control system, which represents an additional step towards beamline coordinated motion.

Once the Delta Tau Power Brick LV is the standard controller for many Sirius' beamlines, the knowledge acquired within the present work can be applied to other systems, *e.g.* implementation of system kinematics in C language for granite bases [11] and the arbitrary trajectory and coordinated motion functionalities for monochromators [12].

ACKNOWLEDGEMENT

The authors would like to gratefully acknowledge the funding by the Brazilian Ministry of Science, Technology, Innovation and Communication and the contribution of the LNLS team, and the Omron DeltaTau team for technical support, providing hexapod code examples. Besides, Diamond Light Source for sharing development knowledge through their open-source repositories.

REFERENCES

[1] L. Liu, M. B. Alves, F. H. de Sá, A. C. S. Oliveira, and X. R. Resende, "Sirius commissioning results and operation status," in *Proceedings of the 12th Int. Particle Accelerator Conf. (IPAC'21)*, Campinas, Brazil: JACoW Publishing, May 2021.

[2] L. Liu, X. Resende, and F. De Sá, "A new optics for sirius," in *Proceedings of the 7th Int. Particle Accelerator Conf. (IPAC'16)*, doi:10.18429/JACoW-IPAC2016-THPMR013, Busan, Korea: JACoW, Jun. 2016, pp. 3413–3416, ISBN: 978-3-95450-147-2. DOI: doi:10.18429/JACoW-IPAC2016-THPMR013. http://jacow.org/ipac2016/papers/thpmr013.pdf

[3] *Bestec p468 mirror unit esm nsls-ii*, Aug. 12, 2021. https://www.bestec-berlin.de/2019/optics/product-category/mirror-unit/3983/p468-mirror-unit-ipe-lnls/

[4] J. .-. Merlet, *Parallel Robots*. Springer-Verlag GmbH, Jul. 2006, ISBN: 9781402041334. https://www.ebook.de/de/product/11430591/j_p_merlet_parallel_robots.html

[5] *Power brick lv ims*, Aug. 12, 2021. https://www.faradaymotioncontrols.co.uk/page/power-brick-lv-ims/

[6] J. J. Craig, *Introduction to robotics: mechanics and control*, 3rd. Upper Saddle River: Pearson Education International, 2005.

[7] T. Kluyver *et al.*, "Jupyter notebooks - a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Scmidt, Eds., IOS Press, 2016, pp. 87–90. https://eprints.soton.ac.uk/403913/

[8] M. Beg *et al.*, "Using jupyter for reproducible scientific workflows," *Computing in Science & Engineering*, vol. 23, no. 2, pp. 36–46, Mar. 2021. DOI: 10.1109/mcse.2021.3052101.

[9] J. Ehiwario, "Comparative study of bisection, newton-raphson and secant methods of root finding problems," *IOSR Journal of Engineering*, vol. 4, pp. 01–07, Apr. 2014. DOI: 10.9790/3021-04410107.

[10] Diamond Light Source, *Arbitrary trajectory github repository*, Sep. 14, 2021. https://github.com/dls-controls/PMAC-Trajectory-Scans

[11] G. N. Kontogiorgos, A. Y. Horita, L. M. Santos, M. A. L. Moraes, and L. F. Segalla, "The mirror systems benches kinematics development for sirius/lnls," presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS), Shanghai, China, Oct. 2021, paper TUPV001.

[12] L. M. dos Santos *et al.*, "The control system of the four-bounce crystal monochromators for sirius/lnls beamlines," presented at the 18th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS), Shanghai, China, Oct. 2021, paper TUPV003.