# QPmR Quasi-polynomial Root-Finder: update and improvements

**Adam Peichl** [*] **Tomas Vyhlidal** [*]

[*] *Department of Instrumentation and Control Engineering, Faculty of Mechanical Engineering, Czech Technical University in Prague, Prague 6, Czechia (adam.peichl,tomas.vyhlidal@fs.cvut.cz)*

**Abstract:** Quasi-Polynomial mapping-based Rootfinder (QPmR) has been designed as a user-friendly Matlab tool for computing and analyzing spectra of Quasi-polynomial roots located in a closed region of the complex plane. As the QPmR uses mapping-based technique for root location, its applicability has some limits concerning the quasi-polynomial order and structural complexity. Within the presented QPmR update, numerical efficiency and memory effectiveness have been improved to increase speed and enhance reliability of computations. Next to Matlab, the updated QPmR algorithm is now also available as an open-source Python package. The use and efficiency of QPmR is demonstrated on examples.

*Keywords:* time-delay systems, quasi-polynomial

The spectrum properties of time-delay systems are infinite dimensional in nature and therefore pose an algorithmic challenge for determining precisely location of roots of associated quasi-polynomials. Two main groups of methods were developed: (i) methods based on discretization of the system solution operator, e.g., software tool DDE-BIFTOOL Engelborghs et al. (2002), (ii) methods based on discretization of the system infinitesimal generator, e.g., software tool TRACE-DDE Breda et al. (2009). Recently, package TDS-CONTROL Appeltans et al. (2022) contains function for reliable computation of all roots of delay systems located in the given right half plane.

The addressed QPmR algorithm was developed to handle a related problem, i.e. locating all roots of a quasi-polynomial in predefined rectangular region of the complex plane. Compared to the above mentioned methods it is mapping based. The QPmR algorithm was introduced in Vyhlidal and Zítek (2009) and numerically optimized in Vyhlídal and Zítek (2014) [1].

QPmR solves a problem of finding all roots of the quasi-polynomial

$$h(s) = \sum_{i=0}^{n} p_i(s) e^{-s\alpha_i} \tag{1}$$

located in a rectangular complex plane sub-region $\mathbb{D} \in \mathbb{C}$ bounded by $\beta_{min} \le \Re(\mathbb{D}) \le \beta_{max}$ and $\omega_{min} \le \Im(\mathbb{D}) \le \omega_{max}$, where $\alpha_i \in \mathbb{R}_0^+$ are unique delays indexed in descending order and $p_i(s) = \sum_{k=0}^{m_i} p_{i,k} s^k$ are the polynomials with real coefficients. Degree of a quasi-polynomial is $d = \max\{m_i : 0 \le i \le n\}$.

The hearth of QPmR root-finding algorithm is the following. Assume $s = \beta + j\omega,\ \beta, \omega \in \mathbb{R},\ j = \sqrt{-1}$ and split complex valued function $h(s)$ into two real valued functions, i.e. $h(\beta + j\omega) = R(\beta, \omega) + jI(\beta, \omega)$, where $R(\beta, \omega) = \Re(h(\beta + j\omega))$ and $I(\beta, \omega) = \Im(h(\beta + j\omega))$. Next, split condition for root location $h(\beta + j\omega) = 0$ into

$$R(\beta, \omega) = 0, \tag{2}$$

$$I(\beta, \omega) = 0. \tag{3}$$

Then the roots are located at the intersections of zero-level curves of surfaces (2) and (3). The algorithm works as follows:

(1) grid step $d_s$ is determined either via heuristic as $d_s = \frac{\pi}{10\alpha_0}$ or by user,
(2) rectangular region $\mathbb{D}$ is discretized via regular dense mesh grid,
(3) function $h(s)$ is evaluated at each grid point and two sets of zero-level curves are obtained for real part (2) and imaginary part (3),
(4) intersection points of real and imaginary zero-level curves are first estimates of locations of root
(5) a numerical method is used to increase precision,
(6) number of roots $N_\varphi$ is validated by application of *argument principle*
$$N_\varphi = \frac{1}{2\pi j} \oint_\varphi \frac{h'(s)}{h(s)} ds,$$
where $\varphi$ is counterclockwise curve enclosing region $\mathbb{D}$,
(7) if necessary, grid step $d_s$ is adapted and algorithm steps from 2 repeated.

## 1. QPMR PYTHON IMPLEMENTATION

Next to MATLAB, root-finding algorithm is now also available as an open source python package [2]. The package is build on *numpy* [3] and *contourpy* [4] python packages.

[*] The presented research was supported by the Czech Science Foundation under the project 24-10301S

[1] QPmR implementation available at `https://control.fs.cvut.cz/en/qpmr`

[2] Source codes accessible at `https://github.com/LockeErasmus/qpmr`

[3] https://numpy.org/

[4] https://github.com/contourpy/contourpy

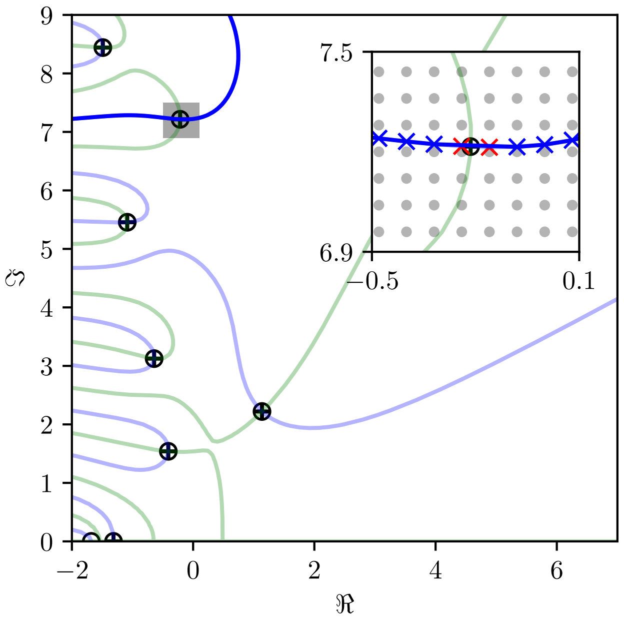

Fig. 1. QPmR algorithm applied to (4), with region (5) and $d_s = 0.08$. Both figures: solid lines zero-level curves (blue - real, green - imaginary), $\oplus$ - location of roots. Detail: grey $\bullet$ - mesh grid, blue x - discretized real zero-level curve, red x - two points used as initial guess for secant method.

As of now, root-finding algorithm supports only quasi-polynomial represented by the table of coefficients and the vector of delays. Even though somewhat restrictive, this allows all operations to be vectorized and therefore computation speed and effectiveness of *numpy* library is fully leveraged. For contour mapping, *contourpy* library is used. Core of this library is written in C++ 11 and allows to obtain and process contours one by one. Library also allows to run in threaded mode and leverage parallel processing of contours.

Memory effectiveness and computation efficiency was increased. For instance, storing the imaginary values evaluated at each point of mesh grid is not necessary. Besides, *secant* method was implemented as an alternative to Newton's method for increasing root precision. While both methods exhibit super-linear convergence, secant method converges slower with order $\frac{1+\sqrt{5}}{2}$ (while Newton has order 2), but at each step, the number of function calls is substantially reduced.

To demonstrate both QPmR algorithm improvements and python package, we consider quasi-polynomial

$$h(s) = \left(1.5s^3 + 0.2s^2 + 20.1\right) + \left(s^3 - 2.1s\right) e^{-1.3s} + 3.2se^{-3.5s} + 1.4e^{-4.3s}, \quad (4)$$

and three sets of region bounds

$$\beta_{min} = -2, \beta_{max} = 7, \omega_{min} = 0, \omega_{max} = 9; \quad (5)$$
$$\beta_{min} = -10, \beta_{max} = 7, \omega_{min} = 0.\omega_{max} = 100, \quad (6)$$
$$\beta_{min} = -10, \beta_{max} = 7, \omega_{min} = 0.\omega_{max} = 200. \quad (7)$$

Algorithm results are depicted in Figures 1 and 2 and performance comparison in Table 1.

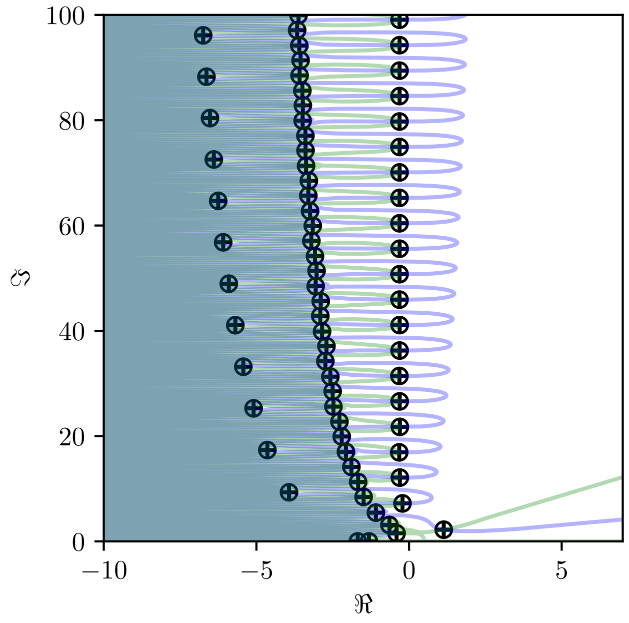

Fig. 2. QPmR algorithm applied to (4), region enlarged to (6). solid lines zero-level curves (blue - real, green - imaginary), $\oplus$ - location of roots.

| | Number of roots | Python $[ms]$ | MATLAB $[ms]$ |
|---|---|---|---|
| region (5) | 8 | $28 \pm 2$ | $42 \pm 4$ |
| region (6) | 71 | $162 \pm 6$ | $250 \pm 20$ |
| region (7) | 139 | $202 \pm 8$ | $1213 \pm 41$ |

Table 1. Performance comparison of Python 3.11 and MATLAB© 2022a QPmR implementations, both run with the same default settings on workstation with AMD Ryzen 5 2600 (6-core, 3.4 GHz, 32 GB RAM), N=1000.

## REFERENCES

Appeltans, P., Silm, H., and Michiels, W. (2022). Tds-control: A matlab package for the analysis and controller-design of time-delay systems. *IFAC-PapersOnLine*, 55(16), 272–277.

Breda, D., Maset, S., and Vermiglio, R. (2009). Trace-dde: a tool for robust analysis and characteristic equations for delay differential equations. In *Topics in Time Delay Systems: Analysis, Algorithms and Control*, 145–155. Springer.

Engelborghs, K., Luzyanina, T., and Roose, D. (2002). Numerical bifurcation analysis of delay differential equations using dde-biftool. *ACM Transactions on Mathematical Software (TOMS)*, 28(1), 1–21.

Vyhlidal, T. and Zítek, P. (2009). Mapping based algorithm for large-scale computation of quasi-polynomial zeros. *IEEE Transactions on Automatic Control*, 54(1), 171–177.

Vyhlídal, T. and Zítek, P. (2014). Qpmr-quasi-polynomial root-finder: Algorithm update and examples. In *Delay Systems: From Theory to Numerics and Applications*, 299–312. Springer.