Rational Matrix Machines

a general algorithm for learning complex rational matrix functions

Anonymous Authors[1]

Abstract

Inferring the rational function that best fits a set of observed outputs at given points is a centuries old problem. Physics often generates resonant phenomena whose mathematical forms are rational functions. They thus appear in myriad systems, from quantum scattering and spectroscopy, to multi-input multi-output (MIMO) transfer functions of coupled linear differential equations in electrical engineering. Because rational functions exhibit both a strong local behavior and a long-range tail effect, they are also used for universal approximation. Since Padé approximants, traditional methods have focused on the polynomials ratio approach, $\boldsymbol{F}(z) \triangleq \frac{\sum_{n=0}^{N_n} \boldsymbol{A_n} z^n}{\sum_{n=0}^{N_d} b_n z^n}$, and requires to pre-specify the degrees of the numerator and denominator N_n, N_d. These methods have been extensively used in physics, electrical engineering, or signal processing.

In this article, we look at this old problem of rational approximation from a novel statistical learning theory perspective. From the partial fraction decomposition form, $\boldsymbol{F}(z) \triangleq \sum_{n=0}^{N_p} \frac{\boldsymbol{R_n}}{z-p_n} + \boldsymbol{C}$, learning these resonances is akin to a one-layer neural network, where the features (or atoms) $\varphi_n(z) \triangleq \frac{1}{z-p_n}$ are determined by the poles $\{p_n\}$, and the linear combination coefficients are the residues $\left\{\boldsymbol{R_n}\right\}$: we call this a *Rational Matrix Machine*.

Training a Rational Matrix Machine (RMM) requires performing both feature extraction (finding the poles $\{p_n\}$) and feature selection (finding the residues $\left\{\boldsymbol{R_n}\right\}$). The former is a highly non-linear process for which current Machine Learning (ML) methods are inadequate.

We here derive a general algorithm to train Rational Matrix Machines (RMM) in a supervised way: given a set of complex points and associated complex matrices $\left\{z_k, \boldsymbol{Y_k}\right\}$ our RMM algorithm learns the complex poles $\{p_n\}$ and matrix residues $\left\{\boldsymbol{R_n}\right\}$ of the best approximate complex matrix-valued rational function $\boldsymbol{F}(z)$, without the need to pre-specify the degrees of the approximation. This establishes a universal ML algorithm for rational functions Hilbert spaces.

Tested on experimental observations of Oxygen-16 nuclear cross sections, our RMM algorithm was able to learn the physical radioactive parameters of the scattering matrix, and match the results predicted by the R-matrix theory of quantum nuclear interactions. This is the first evidence of a ML algorithm learning fundamental physics of nuclear resonances.

1. Introduction – rational matrix functions

DEEP networks have been at the forefront of the monumental progress in machine learning (ML) and artificial intelligence (AI) witnessed in the past decade. However, shallow networks still have their say when addressing certain types of problems. One such one-layer network is formed of rational functions.

Physics often dictates rational functions. Myriad problems give rise to resonant behavior: from transfer functions of coupled linear differential equations in electrical engineering, to physical resonances of quantum scattering interactions in spectroscopy, resonances are intrinsically characterized by a linear combination of complex poles and residues.

Modeling of resonant behavior therefore requires learning rational functions from noisy experimental data: provided a set of observed complex points and associated complex matrices $\left\{z_k\,,\ \boldsymbol{Y_k}\right\} \in \left\{\mathbb{C} \times \mathbb{C}^{h\times q}\right\}_{k\in[\![1,N_k]\!]}$, we must search the Hilbert space of rational functions to infer the one rational function $\boldsymbol{F}(z)$ most likely to have generated the data.

Rational functions are universal approximators (c.f. Runge's 1885 theorem), and rational approximation is a century-old

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

field, pioneered by Heni Padé, which has greatly expanded and been abundantly deployed over the years, either because the form of the problem requires a rational solution, or because of the superior characteristics of rational functions over polynomial fitting, specially their localized and long-tail evanescence properties. For instance "pole-zero modeling" has been central to electrical engineering and signal processing. These traditional rational approximation approaches all focus either on the coefficients of the numerator and denominator, or on its poles and roots (and sometimes on continued fractions):

$$\boldsymbol{F}(z) = \frac{\sum_{n=1}^{N_n} \boldsymbol{A_n} z^n}{\sum_{n=1}^{N_p} b_n z^n} \quad , \quad \boldsymbol{F}(z) = \frac{\bigcirc_{n=1}^{N_n} (\boldsymbol{\alpha_n} - z\mathbf{1} \otimes \mathbf{1})}{\prod_{n=1}^{N_p} (z - p_n)} \tag{1}$$

where $\circ$ designates the element-wise Hadamard product, $\otimes$ the tensor product, and $\mathbf{1} \triangleq [1, 1, \dots, 1]^\mathsf{T}$ is the vector composed of ones.

We here focus on the Hilbert space $\mathcal{F}$ of proper rational matrix functions, i.e. without entire polynomial part other than an offset: $\mathcal{F} = \left\{ \boldsymbol{F} : \mathbb{C} \mapsto \mathbb{C}^{h \times q},\ \boldsymbol{F}(z) \text{ from (1)} \;\middle|\; N_n \leq N_p \right\}$. Using the traditional L_2 norm for regression, solving the rational approximation problem would minimize the distance between the functions in the $\mathcal{F}$ space of rational functions:

$$\begin{gathered} \min_{\boldsymbol{F} \in \mathcal{F}} \mathcal{E}\left[\boldsymbol{F}\right] \\ \mathcal{E}\left[\boldsymbol{F}\right] \triangleq \int_{z \in \mathbb{C}} \left\| \boldsymbol{F}(z) - \boldsymbol{Y}(z) \right\|_{\mathrm{L}_2}^2 \mathrm{d}\rho(z) \end{gathered} \tag{2}$$

In practice, provided observed data points $\left\{ z_k, \boldsymbol{Y_k} \right\}_{k \in [\![1, N_k]\!]}$ with weights $\left\{ \rho_k \right\}_{k \in [\![1, N_k]\!]}$ (so that $\sum_{k=1}^{N_k} \rho_k = 1$), solving the rational approximation problem means minimizing the regularized empirical weighted least square error $\widehat{\mathcal{E}}_\lambda$:

$$\begin{gathered} \min_{\boldsymbol{F} \in \mathcal{F}} \widehat{\mathcal{E}}_\lambda\left[\boldsymbol{F}\right] \\ \widehat{\mathcal{E}}_\lambda\left[\boldsymbol{F}\right] \triangleq \sum_{k=1}^{N_k} \rho_k \left\| \boldsymbol{F}(z_k) - \boldsymbol{Y_k} \right\|_{\mathrm{L}_2}^2 + \mathcal{R}_\lambda\left[\boldsymbol{F}\right] \end{gathered} \tag{3}$$

where $\mathcal{R}_\lambda\left[\boldsymbol{F}\right]$ is a regularization term aimed at compensating the over-fitting tendency of the pure least-squares problem:

$$\begin{gathered} \min_{\boldsymbol{F} \in \mathcal{F}} \widehat{\mathcal{E}}\left[\boldsymbol{F}\right] \\ \widehat{\mathcal{E}}\left[\boldsymbol{F}\right] \triangleq \sum_{k=1}^{N_k} \rho_k \left\| \boldsymbol{F}(z_k) - \boldsymbol{Y_k} \right\|_{\mathrm{L}_2}^2 \end{gathered} \tag{4}$$

Even the latter purely quadratic optimization problem on complex functions is difficult because it is highly non-linear, non-convex, and ill-conditioned. Moreover, the number of poles $N_p \in \mathbb{N}$ (degree of the denominator) is *a priori* unspecified. Yet, previous methods – such as Padé approximation – require explicitly specifying both degrees N_n and N_p, in essence limiting the search to a sub-space of $\mathcal{F}$.

2. Rational Matrix Machines

In this article, we look at this old problem of rational approximation from a novel statistical learning theory viewpoint. Focusing on proper rational functions $\boldsymbol{F} \in \mathcal{F}$, where $N_n \leq N_p$, we take the partial fraction decomposition:

$$\boldsymbol{F}(z) \triangleq \sum_{n=0}^{N_p} \frac{\boldsymbol{R_n}}{z - p_n} + \boldsymbol{C} \tag{5}$$

The rational matrix function $\boldsymbol{F}(z)$ is then akin to a one-layer neural network, where the features (or atoms) $\varphi_n(z)$ are uniquely determined by the poles $\left\{ p_n \right\}$, and the linear combination coefficients are the residues $\left\{ \boldsymbol{R_n} \right\}$:

$$\boldsymbol{F}(z) = \sum_{n=1}^{N_p} \boldsymbol{R_n}\, \varphi_n(z) + \boldsymbol{C} \quad , \quad \varphi_n(z) \triangleq \frac{1}{z - p_n} \tag{6}$$

we shall henceforth call this a *Rational Matrix Machine*.

Mathematically, there is no difference between the Rational Matrix Machine (6) and traditional rational approximations (1). However, they differ in both spirit and form. In form, the Rational Matrix Machine linear combination of features (6) isolates the individual contributions of each resonance, which is often much more convenient for subsequent treatment. In spirit, Rational Matrix Machines portray regression (3) as a machine learning problem: solving (3) means training the Rational Matrix Machine (6). This brings useful statistical learning theory concepts – such as Reproductive Kernel Hilbert Spaces (RKHS) and regularization – to address this centuries-old problem.

Training a Rational Matrix Machine is not straightforward: both feature extraction – learning the number N_p and set of poles $\left\{ p_n \right\}_{n \in [\![1, N_p]\!]}$ – and feature selection – learning the corresponding residues $\left\{ \boldsymbol{R_n} \right\}_{n \in [\![1, N_p]\!]}$ and the offset $\boldsymbol{C}$ – must be performed. Moreover, poles introduce highly non-linear (actually resonant) behavior that impeach resorting to traditional gradient-descent methods for the feature extraction. Thankfully, significant progress has been achieved to converge the poles $\left\{ p_n \right\}$ in the case where their number N_p is fixed: in 2006 Gustavsen implemented a very successful Sanathanan-Koerner iteration by barycentric pole relocation (BPR), giving birth to the now widespread Vector Fitting (VF) algorithm [CITE]. Regarding how to learn the number of poles N_p, problem-specific *ad-hoc* approaches have been proposed to filter out spurious poles [CITE pole trim-

ming], though the authors are unaware of any systematic and general method.

3. Training Rational Matrix Machines

In this article, we build upon these past achievements and establish a universal ML algorithm to train Rational Matrix Machines (RMM). We break down the feature extraction process into iterations of three steps: 1) regularized barycentric residues solving (performed with a new Elastic-Net type of regularization on complex Frobenius norms we specially tailored to Rational Matrix Machines and which we managed to solve by generalizing the Proximal-Forward-Backward-Splitting (PFBS) algorithm to complex matrices, section 3.1.1); 2) pole filtering (which removes superflous poles, section 3.1.2) and; 3) barycentric pole relocation (which updates the poles to convergence, section 3.1.3). Once the features are extracted, the feature selection step becomes a classical regularized least-square problem (section 3.2).

3.1. Feature extraction – pole finding

The process of feature extraction consists of learning the set of poles $\{p_n\}_{n\in[\![1,N_p]\!]}$, including their number N_p. To do so we take a "kitchen sink" approach whereby we start at t_0 with a large number of poles $N_p^{t_0} \gg N_p$, say $N_p^{t_0} = \text{floor}\left[\frac{N_k}{3}\right]$ (it is impossible to infer a resonance from less than 3 points). We then set-up the following iterative scheme to progres-sively diminish the estimated number of poles $\{p_n\}_{n\in[\![1,N_p]\!]}$ until we hopefully converge to the true value N_p.

Finding the poles $\{p_n\}_{n\in[\![1,N_p]\!]}$ in (4) is a highly non-linear problem. RMM elegantly overcome this by decomposing the pole finding problem (feature extraction) into three steps, each solvable by standard optimization or linear algebra methods, and iterating through until the poles converge in both numbers and values.

3.1.1. Regularized barycentric residues solving

For this first feature-extraction step, we fix the poles $\{p_n\}_{n\in[\![1,N_p^t]\!]}$ and find the corresponding residues by solving the following RMM problem (7), where (8) defines a special type of barycentric Elastic-Net regularization for complex rational matrix functions:

$$\min_{\boldsymbol{R_n},\boldsymbol{C},r_n} \widehat{\mathcal{E}}_{\text{RMM}} \tag{7}$$

where

$$\begin{aligned}
\widehat{\mathcal{E}}_{\text{RMM}} &\triangleq \widehat{\mathcal{E}}_{\text{SK}} + \mathcal{T}_\mu + \mathcal{P}_\lambda \\
\widehat{\mathcal{E}}_{\text{SK}} &\triangleq \sum_{k=1}^{N_k} \rho_k \left\| \boldsymbol{F}(z_k) - b_i(z_k)\boldsymbol{Y_k} \right\|_{\text{L}_2}^2 \\
\mathcal{T}_\mu &\triangleq \mu \sum_{n=0}^{N_p} \frac{\|\boldsymbol{R_n}\|_{\text{L}_2}^2}{\nabla_n(p_n)} \\
\mathcal{P}_\lambda &\triangleq \lambda \sum_{n=0}^{N_p} \frac{\|\boldsymbol{R_n}\|_{\text{L}_2}}{\Delta_n(p_n)}
\end{aligned} \tag{8}$$

and were the barycentric function $b_i(z)$ is defined as

$$b_i(z) \triangleq 1 + \sum_{n=1}^{N_p} \frac{r_n^{i+1}}{z - p_n^i} \tag{9}$$

The key is to ensure the introduced barycentric function $b_i(z)$ goes to unity as we iterate ($b_i(z) \underset{i\to\infty}{\longrightarrow} 1$), so that we end-up solving the initial regularized problem (3), albeit only on the residues. This is achieved by enforcing the barycentric function (9) shares the same poles as the rational matrix function at iteration i:

$$\boldsymbol{F}_i^\ell(z) \triangleq \sum_{n=1}^{N_p^t} \frac{\boldsymbol{R_n}^\ell}{z - p_n^i} + \boldsymbol{C}^\ell \tag{10}$$

Sovling the RMM minimization problem (7) can be achieved by fixing the poles $\{p_n\}_{n\in[\![1,N_p^t]\!]}$ and performing the following gradient-descent-type Proximal Forward-Backward Splitting (PFBS) algorithm:

$$\begin{bmatrix} \boldsymbol{R_n}^{\ell+1} \\ \boldsymbol{C}^{\ell+1} \\ r_n^{\ell+1} \end{bmatrix} = \begin{bmatrix} \text{prox}_{\mathcal{P}_\lambda}\left(\boldsymbol{R_n}^\ell - \gamma_{\boldsymbol{R_n}} \cdot \eth_{\boldsymbol{R_n}} \left[\widehat{\mathcal{E}}_{\text{SK}} + \mathcal{T}_\mu\right]^\ell\right) \\ \boldsymbol{C}^\ell - \gamma_{\boldsymbol{C}} \cdot \eth_{\boldsymbol{C}} \widehat{\mathcal{E}}_{\text{SK}}^\ell \\ r_n^\ell - \gamma_{r_n} \cdot \eth_{r_n} \widehat{\mathcal{E}}_{\text{SK}}^\ell \end{bmatrix} \tag{11}$$

where the proximal operator can be expressed as

$$\begin{aligned}
&\text{prox}_{\mathcal{P}_\lambda}\left(\boldsymbol{R_n}^\ell - \gamma_{\boldsymbol{R_n}} \cdot \eth_{\boldsymbol{R_n}} \left[\widehat{\mathcal{E}}_{\text{SK}} + \mathcal{T}_\mu\right]^\ell\right) = \\
&\qquad \left(\boldsymbol{R_n}^\ell - \gamma_{\boldsymbol{R_n}} \cdot \eth_{\boldsymbol{R_n}} \left[\widehat{\mathcal{E}}_{\text{SK}} + \mathcal{T}_\mu\right]^\ell\right) \times \\
&\max\left\{0, 1 - \frac{\lambda/\Delta_n(p_n)}{\left\|\boldsymbol{R_n}^\ell - \gamma_{\boldsymbol{R_n}} \cdot \eth_{\boldsymbol{R_n}} \left[\widehat{\mathcal{E}}_{\text{SK}} + \mathcal{T}_\mu\right]^\ell\right\|_{\text{L}_2}}\right\}
\end{aligned} \tag{12}$$

while the gradients can be expressed as

$$\eth_{\boldsymbol{R_n}} \left[\widehat{\mathcal{E}}_{\text{SK}} + \mathcal{T}_\mu\right]^\ell = 2\sum_{k=1}^{N_k} \frac{\rho_k}{(z_k - p_n^i)^*}\left[\boldsymbol{F}_i^\ell(z_k) - b_i^\ell(z_k)\boldsymbol{Y_k}\right] + 2\mu_i \frac{\boldsymbol{R_n}^\ell}{\nabla_n(p_n)} \tag{13}$$

$$\eth_{\boldsymbol{C}} \widehat{\mathcal{E}}_{\text{SK}}^i \left[\boldsymbol{F}_i^\ell, b_i^\ell\right] = 2\sum_{k=1}^{N_k} \rho_k \left[\boldsymbol{F}_i^\ell(z_k) - b_i^\ell(z_k)\boldsymbol{Y_k}\right] \tag{14}$$

$$\eth_{r_n} \widehat{\mathcal{E}}_{\text{SK}}^i \left[\boldsymbol{F}_i^\ell, b_i^\ell\right] = -2\sum_{k=1}^{N_k} \frac{\rho_k \boldsymbol{Y_k}^\dagger}{(z_k - p_n^i)^*}\left[\boldsymbol{F}_i^\ell(z_k) - b_i^\ell(z_k)\boldsymbol{Y_k}\right] \tag{15}$$

These iterations will converge to the $\boldsymbol{R_n}$, $\boldsymbol{C}$ and r_n that minimize (7) as long as the learning rates satisfy:

$$\gamma_{\boldsymbol{R_n}} \leq \frac{1}{2\sum_{k=1}^{N_k} \frac{\rho_k}{|z_k - p_n^i|^2} + 2\frac{\mu}{\nabla(p_n)}} \tag{16}$$

$$\gamma_{\boldsymbol{C}} \leq \frac{1}{2\sum_{k=1}^{N_k} \rho_k} = \frac{1}{2} \tag{17}$$

$$\gamma_{r_n} \leq \frac{1}{2\max\left\{\mathcal{S}_p\left(\sum_{k=1}^{N_k} \frac{\rho_k}{|z_k - p_n^i|^2} \boldsymbol{Y_k}^\dagger \boldsymbol{Y_k}\right)\right\}} \tag{18}$$

One can initialize the iterations with the solutions to the direct least square system (4), and then descend from there with initial $r_n = 0$.

3.1.2. Pole filtering

As a result of the sparsity-inducing regularizer $\mathcal{P}_\lambda$ in (8), the output of step 1 (Regularized barycentric residue selection, section 3.1.1) will have some residues set to zero $\boldsymbol{R_n} = \boldsymbol{0}$.

This pole filtering step simply consists of removing the poles corresponding to these annulled residues. We thus diminish the number of poles $N_p^t \geq N_p^{t+1}$ by filtering out the superfluous ones, here identified as those which yield residues not robust to small changes in the data (thus most likely over-fitting).

3.1.3. Regularized barycentric pole relocation

With these remaining poles and residues $\left\{p_n^i, r_n^{i+1}\right\}_{n\in[\![1,N_p^{t+1}]\!]}$, we define the new the poles $\left\{p_n^{i+1}\right\}_{n\in[\![1,N_p^{t+1}]\!]}$ as the roots of the barycentric function $b_i(z)$, from which the superflous poles have been removed. That is we solve $b_i(z) = 0$, and update the poles as:

$$b_i(z) = \frac{\prod_{n=1}^{N_p^{t+1}} \left(z - p_n^{i+1}\right)}{\prod_{n=1}^{N_p^{t+1}} \left(z - p_n^i\right)} \tag{19}$$

As poles converge, we will have $b_i \underset{i\to\infty}{\longrightarrow} 1$, entailing we solve the initially intended minimization (3).

To find the next-iteration poles $\left\{p_n^{i+1}\right\}_{n\in[\![1,N_p^t]\!]}$ in (19), solving for the roots of $b^i(z) = 0$ is equivalent to finding the eigenvalues of the matrix $\boldsymbol{Diag}\left[p_n^i\right] - \boldsymbol{r}^{i+1} \otimes \boldsymbol{1}_n^\mathsf{T}$, that is:

$$\left\{p_n^{i+1}\right\}_{n\in[\![1,N_p]\!]} = \mathcal{S}_p\Big(\boldsymbol{Diag}\left[p_n^i\right] - \boldsymbol{r}^{i+1} \otimes \boldsymbol{1}_n^\mathsf{T}\Big) \tag{20}$$

where $\mathcal{S}_p(\,\cdot\,)$ calls the spectrum (eigenvalues), $\boldsymbol{Diag}\left[p_n^i\right]$ is the diagonal matrix of the i-th iteration poles, $\boldsymbol{1}_n$ is the N_p-length vector of ones $\boldsymbol{1}_n = [1, \ldots, 1]^\mathsf{T}$, and $\boldsymbol{r}^{i+1} = \left[r_1^{i+1}, \ldots, r_{N_p}^{i+1}\right]^\mathsf{T}$ is the N_p-length vector of residues of barycentric function $b^i(z)$.

3.1.4. On regularization and meta-parameters

The last remaining thing are the two meta-parameters λ_i and μ_i: λ_i should progressively go from zero to whatever relationship we find with respect to σ the noise, and so should μ_i (which would probably be better it enters as the number of poles stabilizes for a now feature extraction).

Test all pole resolution parameter for optimal regularization:

$$\nabla_n(p_n), \Delta_n(p_n) \triangleq \begin{cases} \Delta_n^1(p_n) & \triangleq & \min_k |z_k - p_n| \\ \Delta_n^2(p_n) & \triangleq & \min_k |z_k - p_n|^2 \\ \Delta_n^3(p_n) & \triangleq & 1/\sum_k \frac{\rho_k}{|z_k - p_n|} \\ \Delta_n^4(p_n) & \triangleq & 1/\sum_k \left(\frac{\rho_k}{|z_k - p_n|}\right)^2 \\ \Delta_n^5(p_n) & \triangleq & 1/\sum_k \frac{\rho_k}{|z_k - p_n|^2} \\ \Delta_n^6(p_n) & \triangleq & 1/\sqrt{\sum_k \frac{\rho_k}{|z_k - p_n|^2}} \\ \Delta_n^7(p_n) & \triangleq & 1/\left(\frac{1}{\Re[p_n]} + \frac{1}{\Im[p_n]}\right) \\ \Delta_n^8(p_n) & \triangleq & 1/\left(\frac{1}{\Re[p_n]} + \frac{1}{\Im[p_n]}\right)^2 \end{cases} \tag{21}$$

3.2. Feature selection – residue finding

Once the poles have been converged through the feature extraction process (section 3.1), the residues can be found (feature selection) by solving again the RMM problem (7), but with different values of the regularization meta-parameters in the Elastic-Net (8). In particular, the sparsity-inducting Pole Filtering regularizer can be removed by setting λ to zero: $\mathcal{P}_{\lambda=0}$. The remaining Tichonov regularizer $\mathcal{T}_\mu$ can then be set to the best possible value to avoid overfitting in the residues. Also, since the poles have successfully been converged, one can set $b_i = 1$ when finding the residues in this last step.

To train a RMM one can thus execute algorithm 1:

Algorithm 1 RMM algorithm

Input: N_k data points $\{z_k, \boldsymbol{Y_k}\}$, with vectorized matrices $\boldsymbol{Y_k}$ of dimension d.
Initialize $N_p^{i_0} = \text{floor}\left[\frac{N_k}{3}\right]$ (assume that need at least 3 points to resolve a resonance).
Initialize poles at peak values,
Initialize regularization meta-parameters at zero $\mu_{i_0} = \lambda_{i_0} = 0$
Feature extraction – pole finding
repeat
  RESIDUES SOLVING (section 3.1.1)
  solve (7) with $\mu_{i_0} = 0$ and increasing λ_i in (8)
  initialize $\boldsymbol{R_n}, r_n$ either at zero or by solving (7) with $\lambda_{i_0} = 0$
  repeat
    RMM PFBS iteration (11)
    $\ell \to \ell + 1$
  until r_n converge (requires converging $\boldsymbol{R_n}$, $\boldsymbol{C}$ too)
  POLE FILTERING (section 3.1.2)
  remove all poles corresponding to zero residues: if $\boldsymbol{R_n} = \boldsymbol{0}$ then remove p_n from poles list ($N_P^{t+1} \geq N_p^t$).

  POLE RELOCATION (section 3.1.3)
  update all other poles through barycentric poles relocation (20)
  update regularization meta-parameters:
  $\lambda_i \to \lambda_{i+1}$
  $\mu_i \to \mu_{i+1}$
until p_n converge
Feature selection – residues finding
solve (7) with $b_i(z_k) = 1$, $\lambda_i = 0$ and regularizing μ_i
Ouput: final list of $\{p_n, \boldsymbol{R_n}\}$

4. Rational Matrix Machines Performance

TOY PROBLEMS:
Benchmark performance vs. VF algorithm and test different regularizors definitions on Vlad's toy-problem, depending on the noise ratio (data noise), points-per-poles (data resolution), etc. These are statistical results.

REAL PROBLEMS:
Oxygen-16 ? Nitrogen ? Carbon ? Gold ? U ? Kr-84, Ti-22, Goal "ML algorithm learns nuclear physics".

OBSERVATIONS:
Vlad's function returns 2 times the number of poles and residues, all complex conjugate from one another. To reproduce the Vlad functions, you can either take the Vlad poles and take the real part of our rational function, or learn a rational function with two times the number of poles Vlad gives us.

DELIVERABLES:

- 1. first task is to benchmark the VF performance: in what range (of signal-to-noise ratio, points-per-pole, and resonance width) does it find the right poles?
  To do this: generate many benchmarks from Vlad with different data quality parameters (i.e. signal-to-noise ratio, points-per-pole, and resonance width), for the entire study, we can set the number of poles to 10, it should not change anything to the overall results.
- 2. Only when we know where does VF in general find the right poles can we use those poles and filter out the superflous ones through RMM regularization.

References