Smart Health 38 (2025) 100600

Contents lists available at ScienceDirect

Smart Health

journal homepage: www.elsevier.com/locate/smhl

TUCUXI-Core: An Open Source generic and efficient computing engine for Model-Informed Precision Dosing

Yann Thoma [a],*, Thierry Buclin [b]

[a] School of Engineering and Management Vaud, HES-SO University of Applied Sciences and Arts Western Switzerland, 1401 Yverdon-les-Bains, Switzerland

[b] Service and Laboratory of Clinical Pharmacology, Department of Medicine and Department of Laboratory Medicine and Pathology, Lausanne University Hospital and University of Lausanne, 1011 Lausanne, Switzerland

ARTICLE INFO

Keywords:
Therapeutic drug monitoring
TDM
Pharmacokinetics
Software
Model-Informed Precision Dosing
MIPD
Bayesian inference

ABSTRACT

The absorption, distribution, metabolism, and excretion of drugs vary from one individual to another. As a result, drug concentration profiles in the blood do not evolve in the same way throughout the population. Measuring drug concentrations and individualizing drug dosages therefore makes it possible to respond to the specific needs of patients, so that their circulating exposure to the drug reaches targets that are both effective in achieving treatment objectives and safe regarding potential toxicity.

In this context, software tools can usefully assist prescribers in the process of adjusting drug dosages based on the therapeutic monitoring of blood concentrations, which is increasingly available in clinical laboratories. Bayesian adaptation based on prior population pharmacokinetic knowledge is the gold standard strategy for this. This article presents the core computing engine of *TUCUXI*, a software package that aims to support clinicians in this activity through a user-friendly graphical interface. We describe its main features and architecture, providing a general overview of this open-source project.

1. Introduction

Precision medicine has been widely championed over the past decade as the most promising initiative in the quest for medical progress. This innovative approach aims to tailor the treatment of diseases by taking into account differences in patients' individual biomedical characteristics, genetic makeup, environmental influences, lifestyle, and personal preferences. It aims to move away from the traditional conception of medical treatment designed for the "average patient" as a one-size-fits-all approach, which may be successful in some patients but not in others. Precision medicine aims at targeting the right treatments for the right patient, at the right time and in the right dose. With regard to the latter, stratification of dosing recommendations based on pharmacogenetic testing has been proposed for some drugs, but a more promising and widely applicable approach is *therapeutic drug monitoring* (TDM), i.e. measuring the circulating concentration of the drug to determine the patient's exposure and modifying the dosing regimen to bring this exposure to an optimal target associated with both the best chance of therapeutic success and the lowest risk of toxicity. This activity has long occupied clinical pharmacologists, since the development of efficient assay methods for lithium, digoxin, and phenytoin in the 1960s (Buclin, Gotta, Fuchs, Widmer, & Aronson, 2012). Since then, an increasing number of cardiac, antiepileptic, psychoactive, anti-infective, anticancer and immunosuppressant agents have been recognized as drugs deserving TDM, according to classical criteria developed for determining whether a given medicinal drug is a candidate for a TDM program (Gross, 2001; Holford

* Corresponding author.
E-mail addresses: yann.thoma@heig-vd.ch (Y. Thoma), thierry.buclin@unil.ch (T. Buclin).

https://doi.org/10.1016/j.smhl.2025.100600
Received 13 October 2023; Received in revised form 27 February 2025; Accepted 24 July 2025
Available online 5 August 2025
2352-6483/© 2025 The Authors. Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

& Buclin, 2012). More recently, the advent of analytical technologies of unprecedented performances and ease of development, in particular liquid chromatography coupled to tandem mass spectrometry (LC-MS/MS), has prompted pharmacologists and other medical specialists to further extend the scope of TDM applications not only to a rapidly growing number of small molecules but also to biological agents. However, the adoption of TDM in the medical community is still limited and slowly progressing. The bottleneck limiting the widespread implementation of TDM is no longer technical difficulties or the time it takes to develop analytical methods, but rather the difficulties of clinicians to interpret measurement results obtained from patients in a rational and medically useful way (Buclin et al., 2012). This is where novel computer-based solutions are expected to facilitate prescribers' move towards precision medicine by providing them with effective, validated, flexible and ergonomic TDM-assistance tools.

The development of computer-assisted TDM can be traced back to the seminal work of Lewis Sheiner, who first applied a Bayesian approach to the interpretation of circulating drug concentration results obtained from individual patients with reference to prior knowledge of the drug's pharmacokinetic (PK) profile and variability in the population. He thus obtained maximum likelihood *a posteriori* estimates of the patient's PK parameters governing the drug disposition in their body, which he used to devise the dosing regimen most appropriate to ensure the attainment of optimal circulating exposure (Sheiner, Rosenberg, & Melmon, 1972a). Since then, models, estimation algorithms, methods and software for the analysis of drug population pharmacokinetics have continuously evolved (Goutelle, Woillard, Neely, and Yamada & Bourguignon, 2022; Guidi, Csajka, & Buclin, 2022). A large number of such analyses have been performed and published to date for many therapeutic drugs, providing a remarkable amount of knowledge describing their average PK parameter values in the population, the variability of these parameters between patients, how given individual patient characteristics (covariates) influence them, and which amount of residual intra-patient variability affects individual time-concentration profiles (lumping together laboratory imprecision, model inaccuracies, and biological fluctuations). Regrettably, these efforts have so far mainly served drug developers and registration authorities, while they are taking longer to be brought to patients receiving drug treatments that are suitable candidates for TDM. However, progress is being made on this front too, and specialists in the field are now almost unanimous in embracing the concept of *Model-Informed Precision Dosing* (MIPD), expected to offer patients the benefit of all the knowledge accumulated regarding the population pharmacokinetics of the drugs they receive, through TDM-based individualization of their dosing (Darwich et al., 2021).

In this regard, it is undisputed that TDM can take advantage of dedicated software tools, as complex computations are generally required for performing state-of-the-art MIPD. Over the last few decades, several tools of this type have been made available, and a few studies inventorying software developed for computer-aided TDM have been published. Del Valle-Moreno, Suarez-Casillas, Mejías-Trueba, Ciudad-Gutiérrez, Guisado-Gil, Gil-Navarro, and Herrera-Hidalgo (2023), Fuchs, Csajka, Thoma, Buclin, and Widmer (2013), Kantasiripitak et al. (2020a, 2020b). They show an increasing interest in MIPD software, but still, it seems that only a small subset of the solutions currently available really reach final users. Moreover, most of the tools available lack user-friendliness and are difficult to use by physicians without specialized training in clinical pharmacology. They accommodate only a limited set of drugs. Some are web-based solutions, with the advantage of easy deployment, but with potential problems of data confidentiality; others are stand-alone software, with the advantage that data is stored on the user's computer, but with the disadvantage of having to install software on an institutional computer. A majority of these tools are distributed under license, with the attendant cost problems for not-always wealthy healthcare institutions.

In this context, we developed *TUCUXI* (Dubovitskaya, Buclin, Schumacher, Aberer, & Thoma, 2017), a standalone open-source software with a graphical user interface (GUI) (see Fig. 1) aimed at supporting prescribing physicians and pharmacologists for the interpretation of drug concentration measurements in blood. It offers an intuitive user-friendly interface with graphs of time-concentration curves along with prediction percentiles and optimal exposure targets, dosage adaptation propositions, report generation, and connectivity with external database systems. It can accommodate any drug for which a reasonably common population pharmacokinetic model and concentration exposure targets are available, to be filled by the user into a drug description file. This software can be executed on any operating system (Windows, MacOS, Linux). As low-end laptops could struggle with intensive computation, special attention has been paid to the computing engine, to maximize the responsiveness of the GUI.

This paper focuses on the computing engine of *TUCUXI*, with a special emphasis on the design choices that enable the code to be remarkably efficient. Some trade-offs did guide the development, in particular to cope with the fact that every medicinal drug has its own population kinetics. Therefore, some parts are hard-coded, while user scripts, allowing adding new medical drugs without recompiling the entire software, can supply some. With that respect, percentile calculation has been the main driver for optimization, as we used a Monte Carlo approach requiring the generation of 10'000 patient time-concentration curves for each percentile graph.

The computing engine has been first developed to be embedded within the *TUCUXI* GUI. However, it is also available through a Command Line Interface (CLI), a Python binding library and a Representational State Transfer (REST) server, to allow for deported computations. All these solutions exploit the XML file format as input and output. As such, the computing engine can easily be connected to and exploited by other medical software applications. The current version of *TUCUXI* is made available on this website: https://www.tucuxi.ch. The source code is distributed as open-source software on GitHub (https://github.com/sotalya/tucuxi-core) under the AGPLv3 license, allowing developers to use it, with the obligation of offering publicly all their code under the same license. A research community called *Sotalya* takes care of its continuing development and spreading and sharing, especially concerning drug models. All the currently available models are accessible as well on the *TUCUXI* website.

The next section presents the core functionalities of the computing engine. Section 3 rapidly presents the software architecture and its interface. Then Section 4, 5 and 6 present respectively predictions, percentiles, and dosage adjustments calculations. Section 7 explains the trade-off between genericity and performances, choices that have been made, and results of experiments concerning computing time. Section 8 shows how the computing engine can be operated directly through text commands. Section 9 gives some information about the verification framework developed to validate the software and the drug models. Section 10 presents various studies related to the implementation of *TUCUXI*. Finally, Section 11 presents our current conclusions and lists the future work.

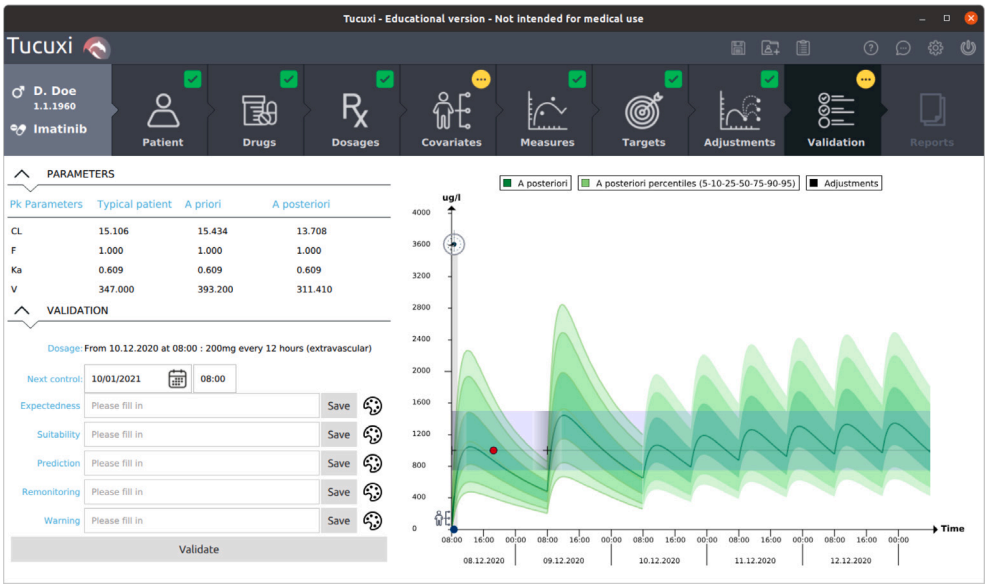

Fig. 1. *TUCUXI* graphical user interface, showing a dosage adaptation.

2. *Tucuxi* core functionalities

TUCUXI core capabilities can be segmented into three main parts:

1. Computation of **concentration predictions** based on *(i)* the patient's dosage regimen, *(ii)* their clinical characteristics known to influence the drug's disposition (i.e. covariates), *(iii)* a set of reference population PK parameters for the concerned drug, and *(iv)* past patient's drug concentration observations, if available;
2. Computation of **prediction percentiles**, based on the same data and exploiting specifically the population variability associated with the drug's PK parameters;
3. Suggestion of **dosage adjustments** aimed at driving the resulting concentration exposure close to the therapeutic targets, specified in the drug description dataset along with clinically usable dosage forms, administration routes, and schedules.

The computations based on the above information about the patient and the drug end up in producing three types of prediction curves and ranges for circulating concentrations:

1. **Typical patient prediction and percentiles**: this corresponds to the concentration curve expected in a *typical patient* having default standard values of every covariate affecting the concerned drug, who would receive the dosage scheme currently administered. The typical curve is surrounded by percentiles that reflect how population variability in the PK parameters combines with intra-patient residual variability to introduce stochastic dispersion in the predictions of concentration levels expected over time. In the GUI version, the percentile levels used by default are 10%, 25%, 50%, 75%, and 90%, but can be modified by the user, and the core computing engine can obviously compute any percentile (with no more computation cost). These percentiles describe the general, non-personalized range of exposure levels susceptible to result from the dosage regimen, bringing an answer to the clinical question: *"Which concentration range would be typically expected in a standard patient under the dosage regimen administered?"*
2. **Prior prediction and percentiles**: this is the concentration curve expected *a priori* in an individual patient defined by their personal values of covariates, under the dosage scheme administered. Compared with the typical patient prediction, this personalized prior prediction integrates the effect of covariates known to affect drug absorption or disposition according to the reference population PK model. Still, it draws no information from the patient's monitoring history. Again, percentiles reflect how PK variability unexplained by covariates combines with intra-patient residual variability to disperse the predictions surrounding the curve. These percentiles materialize the prior distribution used in the Bayesian estimation of the patient's parameters. They are important for the personalized interpretation of monitoring results, as they bring an appropriate answer to the clinical question: *"How do the observed concentrations compare with the levels expected in patients similar to this one under the dosage regimen administered?"*
3. **Posterior prediction and percentiles**: this corresponds to the Bayesian *a posteriori* maximum likelihood concentration curve fitted to the individual patient, taking into account both their specific values of covariates and the concentration measurements obtained during their monitoring history. Compared with the prior prediction, which would apply to any

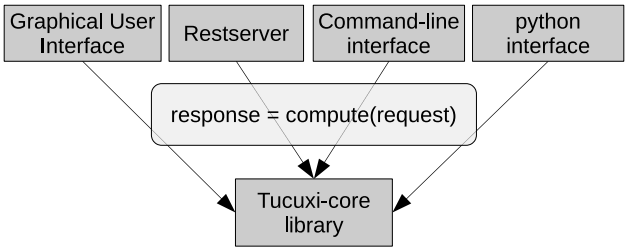

Fig. 2. Software modules and single entry point to the library.

patient having the same set of covariate values, this posterior prediction now individualizes the curve with respect to the concentrations actually observed in the particular patient. Once again, the curve is surrounded by prediction percentiles, which are narrower than the prior percentiles now that the observations have reduced the uncertainty due to the population variability of PK parameters. Should a sufficient number of observations be available for a given patient, their individual parameter values would be considered as known with excellent certainty, and the prediction percentiles for a future observation would shrink down to reflect only the intra-individual residual variability. These percentiles are thus mainly important once decided on the adaptation of the dosing regimen, as they bring an answer to the question: *Which concentration range is expected in this very patient after adjustment of their dosage regimen, taking into account their past monitoring results?"*

Finally, TDM is essentially about adjusting a patient's dosage regimen to bring their concentration exposure close to a target, defined as ensuring the best chances of treatment efficacy with the least risks of toxicity. The information about targets for a monitored drug is drawn from appropriate population pharmacodynamics studies and is contained along with population PK data in the structured *drug model* enabling dosage adjustment for this drug. A set of default targets is defined there and corresponds to the exposure considered optimal in the general population of patients receiving the drug. For some drugs, the target values can be personalized, meaning that they depend on certain patient-specific covariates (e.g. the type of organ transplantation for tacrolimus or cyclosporine, or the degree of sensitivity of a bacterial strain for a given antibiotic). For most drugs, a single target type defines the optimal exposure. However, some drugs are adjusted based on two targets (e.g. peak concentration for efficacy and residual concentration for tolerability of aminoglycoside antibiotics). These dosage adjustments exploit the prediction capabilities of the software and offer a way to propose the best combination of dose, interval, and infusion duration, and as such to bring an answer to the clinical question: *"Which dosage regimen is the best suited to this very specific patient?"*

3. General architecture

First of all, the choice of the programming language has been very important. *TUCUXI*-core has been developed in C++, as this language is one of the best available for performance (Community, 2025; Prechelt, 2000) and modularity. Object-oriented programming, in conjunction with template meta-programming, has been extensively used to create an extremely optimized computing engine.

The core computing engine allows performing predictions on a time range or at specific times, to compute percentiles (typical, prior, and posterior) and to propose dosage adjustments. All these features are offered through a single C++ function, and thanks to object-oriented programming, the input and output data are embedded in a hierarchy of objects. Therefore, integration of *TUCUXI*-core into a software application is done by creating the right input hierarchy, calling a single function, and then analyzing a single output. These input/output object hierarchies are translated from/to XML format, for the CLI, Python binding[1] and REST software. It is therefore straightforward to launch computations from any framework. For C++ developers, of course, it makes sense to integrate the C++ code in order to gain performance.

As illustrated in Fig. 2, this library is currently used by four different software: A command-line interface, the graphical user interface, Python scripts for various studies, and a REST server for remote computations.

Each computation requires some data about the patient:

- Dosage history (doses, route, administration times, infusion duration when applicable)
- Relevant covariates (e.g. age deduced from birthdate, sex, body weight, creatinine clearance, etc.)
- Drug concentration measurements (along with the corresponding sampling times)

Along with these data, specific fields are required, depending on the kind of computation to be done (time ranges for instance). In addition, a description of the population PK model and further characteristics of the drug must be supplied. These models are entered into separate, specifically structured XML files, to allow easy integration of new models as well as sharing models among

[1] Python integration has been done through pybind11, a lightweight library exposing C++ types in Python and vice versa.

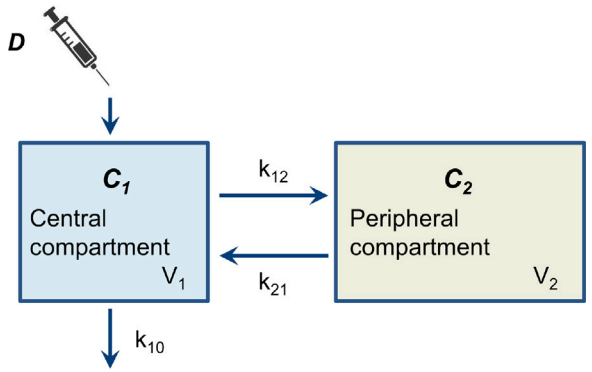

Fig. 3. 2-compartment model.

the community. Therefore, *TUCUXI*-core is generic in the sense that it offers core computing functionalities for various types of PK models, while drug-specific parameter values are supplied thanks to a distinct structure.

The prediction of drug concentration profile over time is the principal functionality of *TUCUXI*-core. Adjacent to this functionality, the prediction of concentrations at specific time points is easily obtained. Percentiles, as detailed in Section 5, exploit the concentration prediction engine to generate 10'000 predictions, which are then used to derive percentiles. In addition, dosage adaptation exploits the software's prediction ability, as it essentially predicts future concentrations for various dosage schemes before proposing the best one.

Therefore, predictions can be considered as the core of the software. While for a single prediction, optimization of the prediction engine does not have a huge impact, when generating 10'000 predictions for percentiles it definitely does in terms of global computation time.

4. Concentration predictions

In classical pharmacokinetics, the description and prediction of drug concentration profiles from reference population data is most often done using compartmental models. Model-free systems have been proposed as well (e.g., Support Vector Machine You, Simalatsar, & De Micheli, 2012) and may be well suited for predicting concentrations at a specific time (typically at the time of measurement), but often suffer from limits of applicability (depending on the method, concentrations below zero or spontaneous concentration increases could be calculated). Model-based approaches are thus better suited to obtain continuous curves. They are used by a majority of pharmacologists, who value their good explanatory power in terms of physiological processes involved in drug disposition.

The basic concept is to model the human body in terms of compartments exchanging quantities of a substance at rates evolving continuously over time. The drug concentration is typically measured in blood plasma, so a central compartment corresponds to the plasma volume along with extravascular volumes of well-perfused tissues, assumed to quickly equilibrate with blood. Models can have more than one compartment, and in this case, either intracellular, fat, or other sets of tissues are considered to slowly upload and release the drug from and to the bloodstream. Drug intake is seen as an input into one of the compartments (usually the central one), and drug elimination corresponds to an irreversible flow of substance leaving the central compartment, either physically through excretion or chemically through metabolic biotransformation. As an example, Fig. 3 represents a 2-compartment model with intravenous bolus injection into the central compartment, drug exchanges between the central and the peripheral compartments, and drug elimination from the central compartment.

Constants k_{10}, k_{12}, k_{21} are used to describe the drug flow rate between the compartments. The underlying pharmacokinetic law states that the probability of transfer of drug particles out of a given compartment is proportional to the number of particles present in that compartment (first order linear process); the model constants are precisely the corresponding coefficients of proportionality. As their corresponding volumes define the compartments, the amounts of drug they contain are expressed as concentrations. In this example, the differential equations governing the model depicted on Fig. 3 are, for concentration C_1 and C_2, respectively in compartments V_1 and V_2:

$$\frac{dC_1}{dt} = k_{21}C_2 - k_{12}C_1 - k_{10}C_1$$
$$\frac{dC_2}{dt} = k_{12}C_1 - k_{21}C_2$$

It is to be noted that in this case, administering a drug consists, for example, in injecting a certain dose D at the time of administration in compartment V_1, and so to add D/V_1 to C_1.

The constants are directly derived from population PK studies, typically performed among a defined population of patients receiving the drug. The equations presented above exploit so-called micro-constants, which can be reparameterized as a set of macro-constants (clearance, volume, etc.) more often used in clinical pharmacokinetics. Depending on the complexity of the differential

equations, integrated analytical equations are available and more efficient in terms of computation time. For linear elimination models up to 3 compartments, such analytical solutions do exist. Conversely, for other models not exclusively following first-order processes, e.g. the relatively simple Michaelis and Menten (1913) single-compartment model assuming saturable drug elimination rate, only non-integrable differential equations are available:

$$\frac{dC}{dt} = \frac{V_{max} \cdot C}{K_M + C}$$

It is to be noted that all those standard analytical equations apply to the description of drug *disposition*, i.e. distribution and elimination. Combining them with further components describing drug *absorption* leads to a variety of models accounting for concentration profiles obtained typically after **intravenous bolus injection**, **continuous rate intravenous infusion** or **extravascular administration** involving specific absorption kinetics. Moreover, the possibility of repeating dose administrations at a given frequency leads to declining the models in three flavors:

(1) **Single dose** without any repetition ;
(2) **Multiple doses**, starting from the beginning of the treatment and leading to a certain degree of accumulation of concentrations;
(3) **Steady state**, when the treatment has sufficiently lasted so that concentrations steadily oscillate around an average plateau between trough and peak values from one dose administration to the next one.

TUCUXI is able to deal with the specification of patient covariates known to affect PK parameters through a given arithmetical relationship with defined coefficients, and even with their modification over time. Doing so means that at certain time points, the PK parameters shall change. In order to deal with this dynamic modification of parameters, and to accept non-linear disposition models, the prediction calculation is based on a fourth flavor (**residual-based**). Each intake corresponds here to the start of a so-called *cycle*, and the residual concentration values in each compartment, retrieved from the previous cycle, are used as initial conditions to compute the current cycle values. This makes the set of available analytical models slightly differ from the standard literature. This architectural choice has also been driven by the fact that some PK models do not have analytical solutions. As such, the only way to calculate concentration predictions is to iterate over time, thanks to the Runge–Kutta 4th order method (RK4). In that context, this fourth flavor allows using the same architecture for both analytical and differential equations. Moreover, the dosage history, while commonly being composed of regular dose administration intervals, can be more esoteric. *TUCUXI*-core allows for instance to describe weekly dosages where the medical drug is taken at fixed times during the week, with a pause on the weekend.

In terms of performance, when an analytical equation exists, it is hard-coded in the source code and offers a better option compared to the RK4 method. For the RK4 implementations, we took advantage of static polymorphism through the curiously recurring template pattern (Bancila, 2020). Thanks to this pattern and to functions inlining, we could optimize the final software binary while keeping an elegant source code. It offers a way to add a new computation model easily, by only describing the differential equations and the dose administration method.

The macro-constants and the dosage history define exactly the predicted concentration curve. The question is then about the way of evaluating these constants. In a published population PK study, the authors usually present average values of the parameters (macro-constants), along with their inter-individual variability and their arithmetical relationship with influential patient covariates. Population PK models also report an intra-individual or residual variability, which lumps together measurement imprecisions, model misfits, and unaccounted biological fluctuations.

Based on this information, *TUCUXI* can first calculate predictions for the so-called *typical patient*. The average parameter values reported in a study correspond to such a typical individual, and if no information is available about a patient, this calculation gives a first approximation of what could be the observed concentration profile based on the drug dosage history.

If the patient's covariates are known, the published model proposes to adapt the average parameters thanks to update functions. This kind of adaptation is referred to as *a priori* calculation and results in a prediction reflecting more precisely the concentrations expected for an individual sharing these characteristics. Typical covariates are age, sex, or weight, but *TUCUXI* can handle any kind and number of covariates defined in a specific model.

For each model, covariates can influence either PK parameters or other covariates. For instance, the creatinine clearance might influence the drug's clearance, but might also be influenced by the weight of the patient. Computationally speaking, a dependency graph is extracted from the description of parameters and covariates and used to calculate the current parameters based on the evolution of covariates.

Finally, the most accurate prediction is made when, in conjunction with covariates, the software "knows" the results of real concentration measurements performed at specific times. The method currently implemented is based on a maximum likelihood function derived from the Bayesian approach. It allows taking into account the population reference values as well as the actual measurements done in the individual patient, by exploiting the intra- and inter-individual variability. The resulting *a posteriori* prediction is the one used for dosage adjustment. The corresponding curve does not pass exactly through the measured concentration values, but rather incorporates a certain degree of regression to the mean.

Often a prediction must be made at steady-state. In this case, the calculator uses the available equations to calculate the residuals of several successive cycles in order to reach the steady state, which then serves as a starting point for the prediction. To ease computation, an approximate time required to reach equilibrium is entered in the drug model. It is interesting to note that for models having an analytical solution it is very fast to calculate the successive residuals, while for models implemented in the form of differential equations, the RK4 method implies a complete calculation of the cycles leading to steady-state.

In terms of performance, all the parameters of the equations that can be shared between the cycles are calculated only once. Likewise, for models implemented in differential equations, templates, and static inheritance have been used to obtain hyper-optimized code. It is clear that for a simple prediction, the impact is not too great, but when generating 10,000 curves, this kind of optimization is crucial.

4.1. A priori parameters

A priori parameters, as previously mentioned, are computed from the patient covariates. Computationally speaking, a TinyJS interpreter (Williams, 2015) is used to run the *a priori* parameters generation. It offers the possibility to write such Javascript-like code within the drug model description without recompiling the entire library. This genericity will be further discussed in Section 7, but it is important to notice that it allows performing any kind of arithmetical computations describing how covariates influence PK parameters.

The following code shows the *a priori* clearance calculation, from a published Imatinib model (Gotta et al., 2012), implemented in a *TUCUXI* drug model. We can distinguish the covariates: `bodyweight`, `age`, `sex`, and `gist`. The age is automatically calculated from the birth date of the patient, `sex` is 1 for male, 0 for female (software convention), and `gist` is a boolean value representing the diagnosis of gastrointestinal stromal tumor. The `theta<n>` are typically the coefficient values extracted from the original publication, and `TVCL` is the individualized clearance calculated.

```
theta1 = 14.3;
theta4 = 5.42;
theta5 = 1.49;
theta6 = -5.81;
theta7 = -0.806;
FBW = (bodyweight - 70) / 70;
FAGE = (age - 50) / 50;
if (gist) PATH = 1; else PATH = 0;
TVCL = theta1 + theta4 * FBW + theta5 * sex - theta5 * (1-sex)
    + theta6 * FAGE + theta7 * PATH - theta7 * (1 - PATH);
return TVCL;
```

This flexibility also comes with an appropriate management of units. *TUCUXI* is aware of certain units, and can automatically convert the input data into the required unit. For instance, in the imatinib example, the body weight has to be entered in Kilograms, and so if the input is in grams it will be converted. The list of available units is available for the drug model writers, and can be augmented if required.

Obviously, entering extreme values for covariates could end up with strange results. Here, for instance, if the age value is too small, the clearance can go negative, which is actually impossible and would result in an unlimited augmentation of drug concentration in blood. Therefore, the drug models embed information about drug domain constraints. They consist of Javascript-like equations responsible for validating the covariate values.

Following the same example, the Imatinib model currently implemented has some constraints on the patient's age, which should be in the range $[20, 88]$:

```
return ((age >= 20) && (age <= 88));
```

Therefore, computation could end up with bad results if these constraints are not met. In the graphical user interface, for instance, a message is displayed to let the user check the patient covariates in case such constraints are not met.

In terms of performance, obviously, using a TinyJS interpreter could have a huge impact, as this code is interpreted at run time. However, the covariates' influence on PK parameters is only computed once per covariate change. This is valid for a single prediction, but also for percentiles computations. Therefore, this gain of genericity does not cause a time penalty for percentiles, compared to the computation of 10'000 predictions.

4.2. A posteriori parameters

While *a priori* parameters are calculated thanks to the patient covariates, *a posteriori* parameters require at least one concentration measurement. A Bayesian approach (Lee, 2012), combined with a search for maximum likelihood allows finding the maximum *a posteriori* estimation (MAP) of PK parameters.

We are here interested in finding the most likely set of parameter values θ_{ind}, taking into account the observed concentration(s) C_{obs} and the *a priori* parameter values. This can be done by maximizing the likelihood of the parameters, with the following equation:

$$\theta_{ind} = \hat{\theta}_{MAP} = arg \underset{\theta}{max}\ L(\theta|C_{obs}) = arg \underset{\theta}{max}\ P(\theta|C_{obs}) \tag{1}$$

Bayes theorem states that:

$$P(\theta_{ind}|C_{obs}) = \frac{P(C_{obs}|\theta_{ind}) \cdot P(\theta_{ind})}{P(C_{obs})} \tag{2}$$

In this equation, $P(C_{obs})$ is constant, and so does not need to be taken into account for the maximization. So, Eq. (1) becomes:

$$\theta_{ind} = arg \max_{\theta} P(C_{obs}|\theta_{ind}) \cdot P(\theta_{ind}) \tag{3}$$

Therefore, we only need to maximize:

$$Pm = P(C_{obs}|\theta_{ind}) \cdot P(\theta_{ind}) \tag{4}$$

Or equivalently to minimize the *objective function*:

$$-logPm = -\log P(C_{obs}|\theta_{ind}) - \log P(\theta_{ind}) \tag{5}$$

Each parameter $\theta_{ind(i)}$ is expressed as an *a priori* average value and a deviation from average $\eta_{(i)}$, where $\eta_{(i)}$ acts on the parameter value with respect to a distribution function. Therefore, maximizing the prior $P(\theta_{ind})$ corresponds to maximizing the likelihood of the parameters. Considering a covariance matrix Ω for the parameters, the log of the prior is:

$$\eta^T \cdot \Omega^{-1} \cdot \eta$$

The log of posterior, $P(C_{obs}|\theta_{ind})$, is computed as

$$\log(sig) + 0.5 \cdot (\frac{eoDiff}{sig})^2$$

Where, sig and $eoDiff$ depend on the residual error model. For instance, if the residual error model is just an additive model characterized by a σ value, then $sig = \sigma$ and $eoDiff = C_{obs} - C_{exp}$, where C_{exp} is the concentration calculated from the θ_{ind}.

If more than one measurement is available, their posterior are summed up. The $-logPm$ function is then minimized within a gradient descent to find the most likely θ_{ind} values.

In terms of variability, the PK parameters distributions can be specified as Normal, LogNormal, Proportional or Logit, and the residual error model can be additive, proportional, or mixed (additive + proportional) (Wang, 2007).

5. Percentiles

While prediction is at the core of *TUCUXI*, percentiles depicting the width of prediction intervals are essential in order to evaluate the expectedness or "normality" of the patient's exposure with respect to the population.

5.1. A priori

A priori percentiles are computed similarly for either the typical patient (i.e. with all covariates set to their default value) or the individual patient defined by their own covariate values. They are calculated thanks to a Monte Carlo simulation that exploits both the intra- and inter-individual variabilities. A particular attention was put on the implementation of this part of the software, as Monte Carlo simulations can be very slow. An in-depth optimization allows calculating useful percentiles in less than a second on a standard PC (when analytical equations are available).

A simulated set of 10'000 random patients is generated, using the parameters variability and the residual error model.

From the parameters correlation matrix $Corr$, a Cholesky decomposition allows finding a matrix L that satisfies:

$$Corr = L^T \cdot L$$

Random values are then generated from a Normal distribution $\mathcal{N}(0, 1)$ and stored into a vector R. Then we use L to generate the η for each patient:

$$\eta = L \times R$$

These η are then used to act on the *a priori* parameters, the influence being dependent on the parameter variability function. For instance, for a parameter following a Normal distribution, its value will be $\theta_{ind}(prior)_i + \eta_i$, while for a parameter following a LogNormal distribution, its value will be $\theta_{ind}(prior)_i \cdot e^{\eta_i}$. Finally, random variability corresponding to residual intra-patient error is added to the concentration values computed based on simulated ϵ, according to the residual error model specified in the population PK model.

The computing engine generates 10'000 single predictions, and then for each time point the predictions are sorted to extract the percentiles. Both the prediction generation and the sorting are multi-threaded, to take advantage of the CPU cores of the machine running the software. Obviously, the more cores the faster the computation, especially because there is no data dependencies and as such, no inter-threads synchronization required.

5.2. A posteriori percentiles

As explained previously, the availability of one or several concentration measurements in a given patient allows the computation of maximum likelihood *a posteriori* parameter values, thus enabling the prediction of a time-concentration curve for this very individual. Simultaneously, it reduces the amount of uncertainty around the concentration curve, as information characterizing the patient accumulates. The *a posteriori percentiles* indicate the range of prediction for a future measurement performed in the same patient. Asymptotically, a very large number of observations would enable to characterize the patient perfectly, thus reducing the uncertainty around their PK parameters to zero. In this situation, the *a posteriori* prediction error would reduce to the intra-patient residual error stated in the population PK model (lumping together biological fluctuations, lab imprecision and model misfit), which is still expected to affect the next measurement.

TUCUXI applies Baye's theorem to estimate a joint posterior distribution of the population PK parameters, namely a conditional distribution of η after having observed one or several observations. This updated distribution is then used the same way as above to generate *a posteriori* percentiles.

The specific problem here is how to estimate and use this posterior distribution of PK parameters to draw appropriate samples for a Monte Carlo simulation. In a first step, *TUCUXI* evaluates the inverse of the negative hessian of $L(\theta|C_{obs})$ in Eq. (1) above, and uses it as a covariance matrix for a multivariate Student's t-distribution with $\nu = 1$ degree of freedom, centered on the estimated values of η. This distribution can be considered as roughly centered around the posterior distribution to be estimated. It is then used to produce a series of samples, which are further filtered using a Sampling-Importance-Resampling (SIR) algorithm, with the probability of keeping the sample being proportional to its likelihood according to both the population PK model and the observations collected. The advantage of such filtering algorithms is that the absolute value of the likelihood needs not to be computed, but only to be known up to a proportionality constant.

6. Dosage adjustment

Dosage adjustment allows *TUCUXI* to propose the best dosage schemes for a specific patient, taking into account their *a posteriori* individual PK parameters and the exposure target defined in the drug description file. Available doses, intervals and infusion times are described as well in the drug model, and used to generate the candidate dosing regimens.

TUCUXI supports the following target types: (1) Trough concentration, (2) Peak concentration, (3) Mean concentration, (4) Area Under the Curve (AUC) within the dosing interval, (5) AUC normalized over 24 h, (5) Cumulative AUC since treatment beginning, (7) Time spent over a Minimum Inhibitory Concentration (MIC), (8) AUC over the MIC, (9) AUC normalized on 24 h over MIC, (10) AUC divided by MIC, (11) AUC normalized on 24 h divided by MIC, (12) Peak concentration divided by MIC, (13) Trough concentration divided by MIC, (14) Fraction of time concentrations remain above the MIC.

A target is defined by its best expected value T_{best}, a minimum T_{lo}, a maximum T_{up}, an inefficacy alarm, and a toxicity alarm. A candidate administration scheme is considered acceptable if the calculated target is within the range $[T_{lo}, T_{up}]$, else it is discarded. The adjustment engine can return only the best candidate, a best candidate for each dosing interval, or all the acceptable candidates.

The exercise of TDM is tantamount to adjusting the dosage regimen so to lead the concentration curve as close as possible to the defined target or targets, using allowed values of unitary doses and dosing intervals. This adjustment must be based on the PK parameter values associated with the best available prediction of the patient's concentration curve under their current dosage regimen: in the absence of any concentration result, if the patient's covariates are not known, we use the typical patient prediction for a standard patient; if the patient's covariates are known, we use the prior prediction. Once one or several concentrations have been measured, a posterior prediction can be calculated and provides the basis for dosage adjustment. Dosage adjustment is elaborated over two steps:

1. First, the software tests *various changes* in both the unit dose and the dosing interval, while only resorting to values allowed for these variables according to the drug model. Each change is followed by extrapolation of the predicted curve to steady state and assessed for appropriateness. The current dosage is considered *acceptable* if it meets all targets defined in terms of upper and lower limits. Otherwise, it is assessed as either *excessive* (if it falls above T_{up}) or *insufficient* (if it falls below T_{lo}). as above. Only acceptable dosing regimens are kept as possibilities of drug adjustment.
2. In the second step, the adjustments kept as acceptable are further tested to see whether a *transient modulation* affecting just the next dose of the new regimen might further improve the adjustment, by shortening the attainment of the new steady state. This is especially important for drugs with a rather long half-life, when a significant amount of time is necessary for either accumulation or de-accumulation. In such situations, simply setting the unitary dose and dosing interval to new values aiming at the targets at steady state may leave the curve away from the targets during several upcoming intervals. Thus, the software will examine the trajectory of the curves associated with possible dosage adjustments, to see whether they meet the targets during the next dosing interval. If this is not the case, transient modulations are tested to see whether they could improve the adjustment:

    - If the current dosage is insufficient, this can be achieved by giving a higher next dose (loading dose) and/or shortening the next interval (as for example, if the adjustment of an oral antibiotic from 200 mg q. 8 h to 600 mg q. 8 h is improved if a first dose of 1000 mg is given in the adjusted regimen).
    - If the current dosage is excessive, this can be achieved by prolonging the time before the next administration (dose-free interval) and/or giving a diminished next dose.

Practically, this amounts to modifying the starting time of the new treatment cycle corresponding to the adjusted regimen (so that it starts either earlier or later than at regular pace), or to introducing a short, single-dose cycle between the current cycle and the next cycle corresponding to the long-term adjustment. *TUCUXI*-core can actually perform this third step or not, depending on the computation options supplied. For instance, in the GUI software, the user can activate or not this second step, and is anyway able to further modify manually this intermediary dosage cycle and to observe the resulting impact on the predicted concentration curve.

6.1. Assigning a score to dosing candidates

An elementary measure of the quality of a dosage adjustment can be defined as its ability to bring the concentration curve as close to the targets as possible, using only the unit doses and dosing intervals allowed according to the drug model, as well as infusion intervals for such administration. As already introduced, *acceptable* dosage adjustments are those making the curve meet the target(s), i.e. it falling between T_{up} and T_{lo} for the time point(s) defined for the drug.

A *score of adjustment quality* is calculated as the average over each target of the expression:

$$Score = 1 - \frac{(\log(T_{predicted}) - \log(T_{best}))^2}{(0.5 \cdot \log(T_{up}) - 0.5 \cdot \log(T_{lo}))^2}$$

It is remotely inspired by the r^2 statistic. The idea is that this score reflects the squared relative departure of the predicted concentration from the target value at the corresponding time, expressed in the scale of the relative radius of the therapeutic interval (i.e. $0.5 \cdot log(T_{up}/T_{lo})$). For example, consider a dosage adjustment for efavirenz that predicts a trough concentration at 1.50 mg/L, while the target is 2 mg/L and the therapeutic interval between 1 and 4 mg/L; then the score for this adjustment would be 0.828. A prediction to 2.66 mg/L would give the same score. A prediction to exactly 2.00 mg/L would give a score of 1. A prediction hitting the border of the therapeutic interval, either 1 or 4 mg/L would give a score of 0. A prediction either lower that 1 mg/L or higher than 4 mg/L would produce a negative score value, meaning that the adjustment is not acceptable.

Now of course, this score quantifies only one aspect of the dosage adjustment. It does not address its feasibility and convenience, which must still be considered (the drug models description embeds available doses, intervals and infusion times), nor the uncertainty associated with the prediction it is based on. This uncertainty can be evaluated from the prediction percentiles associated to the observed concentration value. Indeed, the likelihood of an observation found very distant from the population prediction is poor, and thus the uncertainty of dosage adjustments based on this monitoring is high. The probability of an artifact must be seriously considered in such situations.

7. Genericity and performance

When facing a new software architecture design, two features are usually at stake: genericity and performance. A software or a piece of software should ideally be as generic as possible. Notably, if data can be entered dynamically, the same code base can be used without recompiling for every single case. For *TUCUXI*, this genericity is mainly exposed through the drug model files. These models could be hard-coded, but doing so would require recompiling and producing a new release every time someone changes something in a model. In that context, genericity has a real meaning, letting pharmacologists create new models without any need to access the source code.

However, genericity usually comes with an impact on performance. A hard-coded part of code will always be at least as efficient as a generic one. Therefore, a crucial part of the architecture design has been to determine the appropriate level of genericity.

A drug description file consists in an XML-tagged dataset and embeds all data required for computation, along with data simply offering comments or descriptions of the model. The required information are:

- PK model type and parameters: mean value, variability, and *a priori* computation rules describing covariates influence
- Covariates that are used as modifiers on the PK parameters, targets and available dosages
- Residual error model: type of error model, and associated sigma values
- Therapeutic targets : type and value
- Available dosages : doses, intervals, infusion times (if required)

The core computation (prediction) is described by differential equations, whose parameters are the aforementioned PK parameters. For linear differential equations, analytical equations exist and are implemented, but for some nonlinear models, no such equations exist. In both cases, if the XML drug model file would embed these equations, the software would be extremely generic, but also extremely poor in terms of performances. Therefore, we chose to implement the core equations in C++, specifically for differential equations, as most of the computing time is spent running predictions.

The implication is that a core model has to be implemented within the software before a drug model file can use it. The vast majority of PK models use linear differential equations, and when a fancier model (e.g. Svensson, Aarnoutse, Diacon, Dawson, Gillespie, Boeree, and Simonsson (2018)) is required for a specific drug, then the core needs to be upgraded. In contrast, the influence of covariates on the PK parameters is described in the drug models, as they do not have a huge impact on performance.

In order to assess the effectiveness of the computing engine, we conducted experiments with two very different drug models. The first one, (Gotta et al., 2012), involves imatinib, which shows a linear elimination with one compartment and is computed

Table 1
Computing times for 48 h of predictions, in [ms], for imatinib and rifampicin, at the treatment start, and after one month.

Model	A priori Start [ms]	A priori 1 month [ms]	A posteriori Start [ms]	A posteriori 1 month [ms]
Imatinib	0.20	0.77	0.25	0.89
Rifampicin	0.45	3.81	2'136.45	2'230.80

Table 2
Computing times for 48 h of percentiles, in [ms], for Imatinib, at the treatment start, and after one month.

Nb of cores	A priori Start [ms]	A priori 1 month [ms]	A posteriori Start [ms]	A posteriori 1 month [ms]
1	701	902	958	1658
2	515	530	702	954
4	528	334	619	591
8	295	263	386	422
16	170	193	209	293

Table 3
Computing times for 48 h of percentiles, in [ms], for Rifampicin, at the treatment start, and after one month.

Nb of cores	A priori Start [ms]	A priori 1 month [ms]	A posteriori Start [ms]	A posteriori 1 month [ms]
1	2736	21'492	34'158	52'724
2	1454	12'106	17'339	28'465
4	843	7'610	9'692	16'330
8	482	5'282	5'388	10'495
16	407	5'042	4'742	9'272

thanks to analytical equations. The second one, Svensson et al. (2018), is about rifampicin and is a complex model expressed as a set of differential equations. For both models, we did compute *a priori* and *a posteriori* predictions and percentiles, at two different times: at treatment start, for 2 days, and after one month of treatment, also for 2 days. As the percentiles computation is intensive and multithreaded, we did run the experiments on 1, 2, 4, 8, and 16 CPU cores to illustrate the advantage of using a multi-core CPU. Finally, we computed the average of 10 runs for each of these experiments. The hardware was an Intel Core i7-10700K @ 3.80 GHz, with 16 cores, and 32 GB of RAM. Tables 1–3 summarize the results. We can observe an impressive difference between both models, here especially because the first one offers analytical equations.

8. Command-line interface

The core computing engine allows performing predictions on a time range or at specific times, to compute percentiles (*a priori* and *a posteriori*), and to propose dosage adjustments. All these features are offered though a single C++ function, and thanks to object-oriented programming, the input and output data are embedded in a hierarchy of objects. Therefore, integration of *TUCUXI*-core in a software is done by creating the right input hierarchy, calling the single function, and then analyzing the single output.

Going from an object-oriented C++ design hierarchy to an XML representation is straightforward. Therefore, a command-line tool has been developed to do computations. A single input XML file shall contain all information about the patient and their treatment as well as what computations shall be done. The program outputs another XML file containing the calculated points as well as information about the covariates and PK parameters computed for the calculation.

The very simple interface (one input/one output) allows automating things thanks to the management of XML files. The command-line interface has been extensively used for validation purpose (see Section 9.3). It is also useful for automating retrospective studies, as in Alrahahleh et al. (2024), Hassine et al. (2024), Drennan et al. (2021), Narayan et al. (2021), Patanwala et al. (2022). For instance, Excel sheets extracted from an institutional database are translated to the *TUCUXI* format, computations are started, and reports are generated with the help of Python scripts. As already stated, a Python binding is also available (https://github.com/sotalya/tucuxi-python), enabling faster computations compared to the option of launching the command line software for each computation.

Finally, this versatility allowed the creation of a computation REST backend server, using the same input/output format. Therefore, the computing core engine can be accessed remotely to develop new web or mobile applications.

9. Verification and validation

TUCUXI is being used to suggest dosage adjustments to medical doctors. Therefore, no computation errors are acceptable. If the program crashes because of an internal error, the user will not be especially happy, but at least no patient will be harmed. However, a wrong calculation or an error in a unit translation can have devastating effects on patients' quality of healthcare.

In order to circumvent these issues, a verification plan has been set up, which comprises three main phases: unit testing, fake PK models, and *NONMEM* cross-validation.

9.1. Unit/integration testing

Each part of the computing core has been verified thanks to testing, within the *Googletest* framework. This standard practice in software development allows for ensuring the correct implementation of basic components. We used an approach composed of unit tests and integration tests, exploiting previously tested components in further component testing.

9.2. Fake PK models

The simplest PK models used in practice are composed of linear differential equations, with available analytical equations. In order to simplify validations of the computing core, we designed specific PK models allowing the test team to create tests easily.

For instance, the following PK model has four parameters, A, R, M and S, D being the dose amount.

$$C(t) = \max(0.0, (D + residual \cdot R) \cdot (1 - t \cdot S) \cdot M + A)$$

As parameters can be influenced by covariates, a drug model with a covariate acting on each of the PK parameters has been designed so that each test can set these values. This was particularly important for percentiles computation validation. Thanks to this model, we could act on the variability distribution of parameters, as well as on the residual error model, knowing exactly how the percentiles should look like.

9.3. Core computing engine validation

NONMEM (Beal & Sheiner, 1980; Sheiner, Rosenberg, & Melmon, 1972b) is the *de facto* standard software widely used by pharmacologists for PK population studies. It is extremely flexible, but also very slow. An extensive, systematic cross-validation of *TUCUXI* was performed on a panel of fabricated models mimicking various pharmacokinetic scenarios following oral, bolus, or intravenous administration, with either sparse or rich sampling, over a wide range of population parameter values. The results confirm the excellent consistency of *TUCUXI* with *NONMEM*, and are detailed in (Ravix et al., 2025).

9.4. Drug models NONMEM cross-validation

Verification of the computing core has been conducted thanks to both previously presented approaches. The drug models are described in external files, and shall be validated as well, and here again, we developed a *NONMEM* cross-validation.

It is currently used to compare calculations with computations done within *TUCUXI*. The same information as the one supplied in the *TUCUXI* drug model has to be written in *NONMEM*. It consists in the PK model, PK parameter values, their variability, the influence of covariates, and the residual variability. An efficient structure of *NONMEM* files has been developed, to reduce the redundancy.

The tests to be conducted are described in specific files and are run by Python scripts. These scripts launch *TUCUXI* in command-line mode and collect its results. They also generate data formatted for *NONMEM*, run *NONMEM*, and aggregate results. Both outputs are then compared and a Junit report is generated. The tests apply to predictions (typical patient, *a priori*, and *a posteriori*), and corresponding percentiles. For percentiles, the scripts generate 10000 random parameters, run *NONMEM* on these fake patients, and compute the percentiles. *A posteriori* percentiles and dosage adaptations are however not evaluated, because *NONMEM* does not offer methods for such calculations.

This cross-validation approach is interesting in the sense that one person can write the *TUCUXI* model, another one the *NONMEM* model, and when comparisons do not fit, the point of discrepancy can be found and corrected. This practice is indeed strongly encouraged, to minimize the risk of errors during the implementation of a new model.

10. Current implementation

TUCUXI has now been implemented in several healthcare institutions, mostly academic, where it is currently being evaluated for its clinical usability in patient care. In particular, clinical pharmacologists have used it as a tool to assess the effective potential of MIPD approaches to improve the attainment of target concentration exposure with given drugs, which represents a suitable proxy for the clinical usefulness of this type of software tool. *TUCUXI* has also been included in two recent comparative evaluations of the MIPD software tools currently available (Del Valle-Moreno et al., 2023; Kantasiripitak et al., 2020b).

10.1. Tucuxi use cases

TUCUXI was successfully implemented in the context of intensive care patients for the *a posteriori* estimation of vancomycin AUC, now considered the best exposure index to use during the TDM of this antibiotic (Goutelle et al., 2023). Moreover, 3 published population PK models for vancomycin were coded in the software and compared for their *a priori* and *a posteriori* predictive performances, using 466 concentrations from 188 patients. This systematic comparison was made possible thanks to the automation of *TUCUXI* with Python scripts (Narayan et al., 2021; Patanwala et al., 2022). Similarly, a series of 9 population pharmacokinetic models of vancomycin in neonates were sequentially compared to identify the one having the best performance as Bayesian *prior* for predicting future exposure in a consecutive cohort of 69 neonates (Alrahahleh et al., 2024). Such automation was used as well to retrospectively compare different TDM strategies based on 1 or 2 samples, using various subsets of a series of 512 tobramycin concentrations obtained during 95 treatment courses (Drennan et al., 2021).

A study of piperacillin dosage adjustment, based on a series of 78 consecutive patients providing two or more TDM samples, indicated a better performance in terms of target concentration attainment when combining TDM with MIPD using *TUCUXI* than with empirical TDM without MIPD (83% versus 32%, $p < 0.001$) (Haefliger et al., 2025). Similarly, we established the potential of model-based dosage adjustment with *TUCUXI* to improve cefepime target attainment, compared to actual empirical TDM in a series of 119 patients (Suttels et al., 2022).

A comparison between *TUCUXI* and *BestDose*, another MIPD tool using a non-parametric approach, was performed based on 264 daptomycin TDM measurements obtained in 94 patients. Both software tools performed in good agreement despite the substantial differences in their mathematical background and showed overall comparable results, confirming their usability for precision dosing of daptomycin Heitzmann et al. (2022).

Busulfan is an anticancer agent used to precondition patients for hematopoietic stem cell transplantation, known to require TDM-based dosage adjustment. *TUCUXI* was compared as well with *BestDose* for the TDM of busulfan in a series of 61 pediatric patients. Both software tools performed in good concordance and showed similar predictive performances (Goutelle, Woillard, et al., 2022). The performance of various TDM sampling schedules to accurately predict actual busulfan blood concentration exposure was also evaluated and compared, based on a series of PK simulations derived from actual patient measurements, using the automation of *TUCUXI* Hassine et al. (2024).

A published population PK model of venetoclax, a targeted anticancer agent, was successfully validated for its utilization as *prior* with *TUCUXI* in an independent set of 20 acute leukemia patients providing 93 TDM samples (Philippe et al., 2024).

10.2. Comparative evaluations of tucuxi

A group of Belgian pharmacologists elaborated a set of comparison criteria and applied them to 10 MIPD software tools described in the literature and available for benchmarking. They reported notable differences between their features, user interface design, number and types of drugs handled, user support, quality control, and cost aspects. Improvements remain globally suitable regarding electronic health record integration, standardization, model validation, and available evidence supporting clinical benefits. Compared with its competitors, *TUCUXI* was strong regarding the output report generated and its cost. Computational aspects and user-friendliness were deemed fair. User support was considered weak point (Kantasiripitak et al., 2020b).

More recently, Spanish authors published a scoping review of 19 MIPD tools currently available for supporting patient care. *TUCUXI* was the only tool found simultaneously among the 4 freely accessible ones, the 5 that enable users to add themselves population PK models for further drugs, the 7 that make integration with electronic health records possible, and the 8 considered intuitive and user-friendly by the authors (Del Valle-Moreno et al., 2023).

11. Conclusion

In this paper, we presented the general architecture and features of the *TUCUXI* core computing engine, a MIPD tool dedicated to computer-assisted TDM. Its aim is to enable patients receiving critical treatments with narrow therapeutic margins to benefit from optimal TDM-based dosage adjustments, supported by the best available knowledge of the population pharmacokinetic/pharmacodynamic characteristics of the corresponding drugs.

It is available as open-source software (https://github.com/sotalya/tucuxi-core) under the AGPLv3 license and further developments are expected in the future.

The current source code supports only a single analyte, which is perfectly fine for the vast majority of available pharmacokinetic models. However, some medical drugs expose the patients to an active therapeutic moiety that relies on two analytes. All core components have to be extended to tackle this evolution, and this is part of the current work. Even if the single analyte case is a subset of the multi-analytes one, this new development requires keeping the focus on the preservation of performance for the majority of uses.

Another research axis consists in the development of a clinical decision support system (CDSS) that should be able to automatically generate a report with potential dosage adjustments, based on all the information available about the patient. This feature would be useful, especially in areas where access to specialists in clinical pharmacology is limited. It is currently underway for the special case of rifampicin (an anti-tuberculosis drug) in Tanzania (Thoma et al., 2024).

Overall, the open source aspect of the project makes it a good candidate for research and implementation of new capabilities, and the community of users/developers is strongly encouraged to actively add novel features to the existing source code.

CRediT authorship contribution statement

Yann Thoma: Writing – original draft. **Thierry Buclin:** Writing – original draft.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yann Thoma reports financial support and article publishing charges were provided by School of Engineering and Management. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The development of *TUCUXI* implied a long list of people and has been started under the ISyPeM 2 project, funded by NanoTera.ch. The authors would like to especially thank Robert Hooper, Bastien Noverraz, Yannis Jeannotat, Yann Sutter, Severine Petitprez, Aziz Chaouch, Chantal Csajka, Monia Guidi, Nicolas Widmer, Aline Fuchs, Anne Ravix, François Veuve, Paul Thoueille, Pascal André, Catia Marzolini, and Laurent Decosterd for their implication and support in the development of *TUCUXI*.

Data availability

No data was used for the research described in the article.

References

Alrahahleh, D., Thoma, Y., Van Daele, R., Nguyen, T., Halena, S., Luig, M., et al. (2024). Bayesian vancomycin model selection for therapeutic drug monitoring in neonates. *Clinical Pharmacokinetics*, *63*(3), 367–380. http://dx.doi.org/10.1007/s40262-024-01353-8.

Bancila, M. (2020). *Modern C++ programming cookbook* (second ed.). O'Reilly.

Beal, S., & Sheiner, L. (1980). The NONMEM system. *The American Statistician*, *34*(2), 118–119. http://dx.doi.org/10.2307/2684123.

Ben Hassine, K., Daali, Y., Gloor, Y., Nava, T., Théorêt, Y., Krajinovic, M., et al. (2024). Simulation-based optimization of sampling schedules for model-informed precision dosing of once-daily and 4-times-daily busulfan in pediatric patients. *Therapeutic Drug Monitoring*, *46*(6), 786–796. http://dx.doi.org/10.1097/FTD.0000000000001217.

Buclin, T., Gotta, V., Fuchs, A., Widmer, N., & Aronson, J. (2012). Monitoring drug therapy. *British Journal of Clinical Pharmacology*, *73*(6), 917–923. http://dx.doi.org/10.1111/j.1365-2125.2012.04237.x.

Community (2025). URL: https://github.com/kostya/benchmarks. Kostya benchmarks.

Darwich, A. S., Aronson, J. K., Ogungbenro, K., Wright, D. F., Achour, B., Reny, J.-L., et al. (2021). Model-informed precision dosing: Background, requirements, validation, implementation, and forward trajectory of individualizing drug therapy. *Annual Review of Pharmacology and Toxicology*, *61*, 225–245. http://dx.doi.org/10.1146/annurev-pharmtox-033020-113257.

Del Valle-Moreno, P., Suarez-Casillas, P., Mejías-Trueba, M., Ciudad-Gutiérrez, P., Guisado-Gil, A. B., Gil-Navarro, M. V., et al. (2023). Model-informed precision dosing software tools for dosage regimen individualization: A scoping review. *Pharmaceutics*, *15*(7), 1859. http://dx.doi.org/10.3390/pharmaceutics15071859, URL: https://www.mdpi.com/1999-4923/15/7/1859.

Drennan, P. G., Thoma, Y., Barry, L., Matthey, J., Sivam, S., & van Hal, S. J. (2021). Bayesian forecasting for intravenous tobramycin dosing in adults with cystic fibrosis using one versus two serum concentrations in a dosing interval. *Therapeutic Drug Monitoring*, *43*(4), http://dx.doi.org/10.1097/FTD.0000000000000900.

Dubovitskaya, A., Buclin, T., Schumacher, M., Aberer, K., & Thoma, Y. (2017). TUCUXI: An intelligent system for personalized medicine from individualization of treatments to research databases and back. In *Proceedings of the 8th ACM international conference on bioinformatics, computational biology,and health informatics* (pp. 223–232). New York, NY, USA: ACM, http://dx.doi.org/10.1145/3107411.3107439.

Fuchs, A., Csajka, C., Thoma, Y., Buclin, T., & Widmer, N. (2013). Benchmarking therapeutic drug monitoring software: a review of available computer tools. *Clinical Pharmacokinetics*, *52*(1), 9–22. http://dx.doi.org/10.1007/s40262-012-0020-y.

Gotta, V., Widmer, N., Montemurro, M., Leyvraz, S., Haouala, A., Decosterd, L. A., et al. (2012). Therapeutic drug monitoring of imatinib. *Clinical Pharmacokinetics*, *51*(3), 187–201. http://dx.doi.org/10.2165/11596990-000000000-00000.

Goutelle, S., Thoma, Y., Buffet, R., Philippe, M., Buclin, T., Guidi, M., et al. (2022). Implementation and cross-validation of a pharmacokinetic model for precision dosing of busulfan in hematopoietic stem cell transplanted children. *Pharmaceutics*, *14*(10), 2107. http://dx.doi.org/10.3390/pharmaceutics14102107.

Goutelle, S., Wallet, F., Thoma, Y., Peclard, J.-R., Bourguignon, L., Cohen, S., et al. (2023). AUC-based monitoring and model-informed precision dosing of vancomycin in critically ill patients: why and how? *Anaesthesia Critical Care and Pain Medicine*, *42*(6), 2352–5568. http://dx.doi.org/10.1016/j.accpm.2023.101286.

Goutelle, S., Woillard, J.-B., Neely, M., Yamada, W., & Bourguignon, L. (2022). Nonparametric approaches in population pharmacokinetics. *Journal of Clinical Pharmacology*, *62*(2), 142–157. http://dx.doi.org/10.1002/jcph.1650.

Gross, A. S. (2001). Best practice in therapeutic drug monitoring. *British Journal of Clinical Pharmacology*, *52*(Suppl. 1), 5S–10S. http://dx.doi.org/10.1046/j.1365-2125.2001.0520s1005.x.

Guidi, M., Csajka, C., & Buclin, T. (2022). Parametric approaches in population pharmacokinetics. *Journal of Clinical Pharmacology*, *62*(2), 125–141. http://dx.doi.org/10.1002/jcph.1633.

Haefliger, D., Mina, L., Guidi, M., Marzolini, C., Thoueille, P., Rothuizen, L. E., et al. (2025). Individualization of piperacillin dosage based on therapeutic drug monitoring with or without model-informed precision dosing: a scenario analysis. *Journal of Antimicrobial Chemotherapy*, *80*(3), 840–847. http://dx.doi.org/10.1093/jac/dkaf007.

Heitzmann, J., Thoma, Y., Bricca, R., Gagnieu, M.-C., Leclerc, V., Roux, S., et al. (2022). Implementation and comparison of two pharmacometric tools for model-based therapeutic drug monitoring and precision dosing of daptomycin. *Pharmaceutics*, *14*(1), 114. http://dx.doi.org/10.3390/pharmaceutics14010114.

Holford, N. H. G., & Buclin, T. (2012). Safe and effective variability-a criterion for dose individualization. *Therapeutic Drug Monitoring*, *34*(5), 565–568. http://dx.doi.org/10.1097/FTD.0b013e31826aabc3.

Kantasiripitak, W., Van Daele, R., Gijsen, M., Ferrante, M., Spriet, I., & Dreesen, E. (2020a). Software tools for model-informed precision dosing: How well do they satisfy the needs? *Frontiers in Pharmacology*, *11*, http://dx.doi.org/10.3389/fphar.2020.00620.

Kantasiripitak, W., Van Daele, R., Gijsen, M., Ferrante, M., Spriet, I., & Dreesen, E. (2020b). Software tools for model-informed precision dosing: How well do they satisfy the needs? *Frontiers in Pharmacology*, *11*, 620. http://dx.doi.org/10.3389/fphar.2020.00620.

Lee, P. M. (2012). *Bayesian statistics: An introduction*. Wiley.

Michaelis, L., & Menten, M. L. (1913). Die kinetik der invertinwirkung. *Biochemische Zeitschrift*, *49*, 333–369.

Narayan, S. W., Thoma, Y., Drennan, P. G., Yejin Kim, H., Alffenaar, J.-W., Van Hal, S., et al. (2021). Predictive performance of Bayesian vancomycin monitoring in the critically ill. *Critical Care Medicine*, *49*(10), 952–960. http://dx.doi.org/10.1097/CCM.0000000000005062.

Patanwala, A. E., Spremo, D., Jeon, M., Thoma, Y., C. Alffenaa, J.-W., & Stocker, S. (2022). Discrepancies between Bayesian vancomycin models can affect clinical decisions in the critically ill. *Critical Care Research and Practice*, *2022*, Article 7011376. http://dx.doi.org/10.1155/2022/7011376.

Philippe, M., Guitton, J., Goutelle, S., Thoma, Y., Favier, B., Chtiba, N., et al. (2024). Pharmacokinetic consideration of venetoclax in acute myeloid leukemia patients: A potential candidate for tdm? A short communication. *Therapeutic Drug Monitoring*, *46*(1), 127–131. http://dx.doi.org/10.1097/FTD.0000000000001151.

Prechelt, L. (2000). An empirical comparison of seven programming languages. *Computer*, *33*(10), 23–29. http://dx.doi.org/10.1109/2.876288.

Ravix, A., Cathignol, A. E., Buclin, T., Csajka, C., Guidi1, M., & Thoma, Y. (2025). Numerical validation of tucuxi, a promising new Bayesian adaptation tool for model-informed precision dosing. *CPT: Pharmacometrics and Systems Pharmacology*, 1–9. http://dx.doi.org/10.1002/psp4.70077.

Sheiner, L. B., Rosenberg, B., & Melmon, K. L. (1972a). Modelling of individual pharmacokinetics for computer-aided drug dosage. *Computers and Biomedical Research*, *5*(5), 411–459. http://dx.doi.org/10.1016/0010-4809(72)90051-1.

Sheiner, L. B., Rosenberg, B., & Melmon, K. L. (1972b). Modelling of individual pharmacokinetics for computer-aided drug dosage. *Computers and Biomedical Research*, *5*(5), 441–459. http://dx.doi.org/10.1016/0010-4809(72)90051-1.

Suttels, V., André, P., Thoma, Y., Veuve, F., Decosterd, L., Guery, B. t., et al. (2022). Therapeutic drug monitoring of cefepime in a non-critically ill population: retrospective assessment and potential role for model-based dosing. *JAC-Antimicrobial Resistance*, *4*(2), dlac043. http://dx.doi.org/10.1093/jacamr/dlac043.

Svensson, R. J., Aarnoutse, R. E., Diacon, A. H., Dawson, R., Gillespie, S. H., Boeree, M. J., et al. (2018). A population pharmacokinetic model incorporating saturable pharmacokinetics and autoinduction for high rifampicin doses. *Clinical Pharmacology and Therapeutics*, *103*(4), 674–683. http://dx.doi.org/10.1002/cpt.778.

Thoma, Y., Cathignol, A. E., Pétermann, Y. J., Sariko, M. L., Said, B., Csajka, C., et al. (2024). Toward a clinical decision support system for monitoring therapeutic antituberculosis medical drugs in tanzania (project TuberXpert): Protocol for an algorithm' development and implementation. *JMIR Res Protoc*, *13*, Article e58720. http://dx.doi.org/10.2196/58720.

Wang, Y. (2007). Derivation of various NONMEM estimation methods. *Journal of Pharmacokinetics and Pharmacodynamics*, *34*(5), 575–593. http://dx.doi.org/10.1007/s10928-007-9060-6.

Williams, G. (2015). URL: https://github.com/gamedevtech/tiny-js-1. TinyJS.

You, W., Simalatsar, A., & De Micheli, G. (2012). RANSAC-based enhancement in drug concentration prediction using support vector machine. In *Proceedings of the international workshop on innovative simulation for healthcare*.