Computer Physics Communications 278 (2022) 108416

Contents lists available at ScienceDirect

Computer Physics Communications

www.elsevier.com/locate/cpc

pICNIK: A python package with isoconversional computations for non-isothermal kinetics ☆,☆☆

Erick Ramírez [a], Sergio Hernández-López [a], Enelio Torres-Garcia [b], Karla Reyes-Morales [a], Jorge Balmaseda [a,*]

[a] Departamento de Polímeros, Instituto de Investigaciones en Materiales, Universidad Nacional Autónoma de México, Ciudad de México 04510, Mexico
[b] Instituto Mexicano del Petróleo, Eje Central Lázaro Cárdenas Norte, # 152, CDMX, 07730, Mexico

ARTICLE INFO

Article history:
Received 21 March 2022
Received in revised form 30 April 2022
Accepted 15 May 2022
Available online 25 May 2022

Keywords:
Python
Isoconversional computations
Non-isotermal kinetics

ABSTRACT

Isoconversional computations are widely-used methods to determine activation energies for thermally stimulated processes. pICNIK (python Isoconversional Computations for Non-Isothermal Kinetics) is an open-source module designed with the purpose of being a seed to a complete Python package to facilitate kinetic computations. It is object oriented with two classes: `DataExtraction` and `ActivationEnergy`. Five isoconversional methods were implemented: Friedmann's method, the Kissinger-Akahira-Sunose method, Ozawa-Flynn-Wall method, Vyazovkin and advanced Vyazovkin. This module allows to compute the activation energy from thermogravimetric data in minutes and the results can be exported as spreadsheet format or comma separated values files instead of traditional tedious and time consuming data processing. The module was validated with simulated data and two study cases: vaporization of n-decane and the thermal degradation of polypropylene.

Program summary
Program Title: pICNIK
CPC Library link to program files: https://doi.org/10.17632/dpwtnmjpj5.1
Developer's repository link: https://github.com/ErickErock/pICNIK
Code Ocean capsule: https://codeocean.com/capsule/0740039
Licensing provisions: MIT
Programming language: Python
Nature of problem: Numerical determination of the parameters of the Arrhenius function and the reaction model under the isoconversion hypothesis.
Solution method: A Python module with implemented isoconversional methods. The algorithms implemented include several computations such as linear regressions, numerical integration and differentiation, and minimizing functions, all over several data sets.
Additional comments including restrictions and unusual features: The initial data processing is focused on thermogravimetry and may not work for calculating conversion to other properties. However, the object-oriented implementation allows to overcome this limitation easily. Methods for calculating the Arrhenius pre-exponential factor and the reaction model remain to be implemented. It is a small module conceived to be further developed by the community. Another limitation of the code is that it does not include a graphical interface.

☆ The review of this paper was arranged by Prof. N.S. Scott.
☆☆ This paper and its associated computer program are available via the Computer Physics Communications homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: balmaseda@comunidad.unam.mx (J. Balmaseda).

https://doi.org/10.1016/j.cpc.2022.108416
0010-4655/

1. Introduction

The aim of any kinetic study is to describe the dependence of the process rate as a function of temperature, through the so-called Arrhenius parameters (activation energy, pre-exponential factor) and the mechanism or model of reaction. Among kinetic analysis techniques thermal analysis is usually applied in studying solid state kinetics. Thermally stimulated processes on solids can be very complex due to multiple processes occurring (multiple reactions, phase changes, mass and heat transfer, *etc.*). Activation energies that vary with evolution of the process are frequently found because of this complexity. In relation with that fact, the International Confederation for Thermal Analysis and Calorimetry (ICTAC) showed in a Kinetic Project that isoconversional methods are suitable for kinetic analysis of multi-step processes [1].

Isoconversional methods are broadly used to describe the phenomenological thermokinetic behavior of different materials. This description can be used for several industrial applications that goes from decision-making on the thermochemical transformation of waste into value-added products, energy or chemical precursors, to shelf-life predictions of drug formulations [2–10]. However, the implementation of isoconversional computations involves extensive numerical computing. Most of the numerical computation focuses on: linear regression, integration or differentiation and optimization; on extensive experimental data. This high computational demand is time consuming and constitutes the bottleneck of thermokinetic analysis. Even though there are commercial and free software available that makes this procedure a lot easier [11], these are limited to: a few isoconversional methods, reaction order kinetics, the available economic resources or an operating system.

Now, programming is a tool for everyday scientific work and there are diverse programming languages capable of solving numerical problems quickly and efficiently. However, depending on the language, this can be a major challenge for those not so familiar with programming. Python offers an interactive environment, not only for developing but also for working with data along with an intuitive vocabulary and syntax. However, no package in the python package index (PyPI) [12] performs isoconversional computations. An Object-Oriented Open-Source module will allow the scientific community to use it as building-block for continuous and discretionary development of complete Python package for kinetic analysis.

The aim of this work is to develop an open-source Python module to perform isoconversional computations for non-isothermal kinetics. This will optimize the data handling stage and thus facilitate the decision-making about the obtained results and following procedures.

2. Theoretical basis

To study the kinetics of thermally stimulated processes the so-called general equation is used, being a function separable into a thermal contribution and a pseudo-mechanistic contribution [13–15]:

$$\frac{\mathrm{d}\alpha}{\mathrm{d}t} = k(T)\, f(\alpha) \tag{1}$$

The thermal contribution is usually modeled as the Arrhenius equation

$$k(T) = A \exp\left[-\frac{E}{RT(t)}\right] \tag{2}$$

where: E is the activation energy, A the Arrhenius pre-exponential factor, or frequency factor, R is the universal gas constant ($8.314\ \mathrm{J\,mol^{-1}\,K^{-1}}$) and $T(t)$ is the instantaneous temperature of the system.

The pseudo-mechanistic contribution, $f(\alpha)$, is a function related to the mechanism of the process and depends on the variable α which measures the evolution of the process as the conversion or fraction that reacted. It is defined as:

$$\alpha = \frac{\zeta_0 - \zeta(t)}{\zeta_0 - \zeta_f} \tag{3}$$

where: ζ is a physical property of the sample that can be monitored under controlled temperature conditions, like mass in thermogravimetric analysis (TGA).

Now, the goal of thermal kinetics is to find the kinetic parameters: E, A and $f(\alpha)$. To achieve that, the kinetic analysis can be performed under isothermal or under non-isothermal conditions. The non-isothermal kinetic analysis is divided into two categories: differential and integral, in both methods kinetic analysis can be done by either, running experiments at a single heating program or running experiments at several heating programs.

If the experiments are run at a single heating program, the analysis is usually focused on proposing the reaction model, $f(\alpha)$, and thus adjusting the experimental data to one of several semi-empirical models that can be found in literature [16], giving the most appropriate A and E. The limitation of this method is that it usually leads to great uncertainty by giving good agreement for different models with very different activation energies and thus little reliability on the results. In the multiple heating programs analysis the aim is to parameterize the reaction rate by means of an effective activation energy.

The ICTAC has published a series of recommendations for the different kinetic analysis [14,15] and also has validated and suggested the use of isoconversional methods for kinetic determinations involving multi-step processes [1].

2.1. Isoconversional methods

These methods require non-isothermal data from multiple heating programs. Isoconversional methods have their name after the homonym principle, which states that at constant conversion degree, the conversion rate depends only on the temperature (see equation (4) and Fig. 1).

$$\left.\frac{\partial \ln(\mathrm{d}\alpha/\mathrm{d}t)}{\partial T^{-1}}\right|_{\alpha} = -\frac{E_\alpha}{R} \tag{4}$$

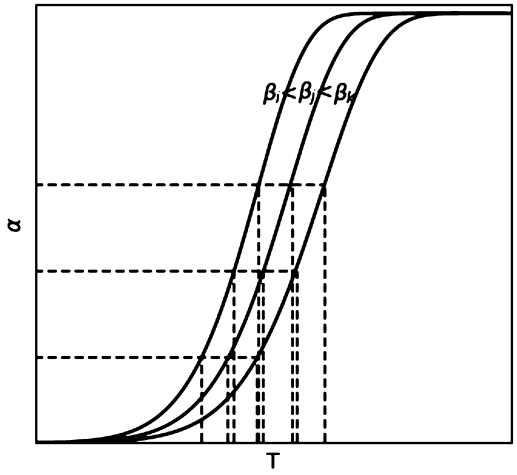


Fig. 1. Illustration of the isoconversion concept. For each conversion value there are n temperature values, where n is the number of heating programs used.

2.1.1. Differential isoconversional method

Friedman's (Fr) method is the most widely used differential method, and its derivation its straightforward by linearizing equation (1) through natural logarithm, taking $k(T)$ according to equation (2) [17]:

$$\ln\left(\frac{\mathrm{d}\alpha}{\mathrm{d}t}\right)\bigg|_{\alpha,i} = \ln\left[A_\alpha f(\alpha)\right] - \frac{E_\alpha}{RT_{\alpha,i}} \tag{5}$$

The i subindex is used to represent the i^{th} temperature program. The activation energy is computed from the slope of a linear regression of $\ln(\mathrm{d}\alpha/\mathrm{d}t)$ *vs.* T^{-1} for each evaluated isoconversion value. A conversion interval from 0.05 to 0.95 is suggested [14].

2.1.2. Integral isoconversional methods

Separating variables in the equation (1) and integrating it, the integral form of the rate equation is obtained:

$$\int_0^\alpha \frac{\mathrm{d}\alpha}{f(\alpha)} \equiv g(\alpha) = A\int_{t_0}^{t} \exp\left[-\frac{E}{RT(t)}\right]\mathrm{d}t = AJ[E, T(t)] \tag{6}$$

which is valid for any heating program. If a linear heating program is considered ($\beta \equiv \mathrm{d}T/\mathrm{d}t = \text{constant}$), then equation (6) turns into:

$$g(\alpha) = \frac{A}{\beta}\int_{T_0}^{T} \exp\left(-\frac{E}{RT}\right)\mathrm{d}T = \frac{A}{\beta}I(E,T) \tag{7}$$

The integral isoconversional methods can be classified based on how they solve the integral $I(E,T)$. By using Doyle's approximation [18], Ozawa [19] and Flynn and Wall [20] (OFW) derived the linear expression:

$$\ln(\beta_i) = \left[\ln\left(\frac{A_\alpha E_\alpha}{g(\alpha)R}\right) - 5.331\right] - 1.052\frac{E_\alpha}{RT_{\alpha,i}} \tag{8}$$

The Kissinger-Akahira-Sunose [21] (KAS) method consists on a similar expression with the Coats-Redfern [22] approximation:

$$\ln\left(\frac{\beta_i}{T_{\alpha,i}^2}\right) = \ln\left(\frac{A_\alpha E_\alpha}{g(\alpha)R}\right) - \frac{E_\alpha}{RT_{\alpha,i}} \tag{9}$$

The activation energy for a given conversion is obtained by a linear regression of $\ln(\beta_i)$ *vs.* ${T_{\alpha,i}}^{-1}$ or $\ln\left(\beta_i/T_{\alpha,i}^2\right)$ *vs.* ${T_{\alpha,i}}^{-1}$ depending on the chosen method.

Another way to obtain the activation energy was proposed by Vyazovkin [23] (V1), by considering $g(\alpha)$ constant at isoconversional conditions the next equation can be obtained:

$$\Omega(E_\alpha) = \sum_{i}^{n}\sum_{j\neq i}^{n-1}\frac{J\left[E_\alpha, T(t)_{\alpha,i}\right]}{J\left[E_\alpha, T(t)_{\alpha,j}\right]} \tag{10}$$

or, for a linear heating rate:

$$\Omega(E_\alpha) = \sum_{i}^{n}\sum_{j\neq i}^{n-1}\frac{\beta_j I\left[E_\alpha, T_{\alpha,i}\right]}{\beta_i I\left(E_\alpha, T_{\alpha,j}\right)} \tag{11}$$

In this case, the integral $I\left[E_\alpha, T_{\alpha,i}\right]$ can be solved numerically or using the Senum-Yang approximation [24]:

$$I(E,T) \approx \frac{E}{R} p(x) = \frac{E}{R}\left[\frac{\exp(-x)}{x}\frac{x^3+18x^2+88x+96}{x^4+20x^3+120x^2+240x+120}\right] \tag{12}$$

with $x = E/(RT)$.

Shortly after, Vyazovkin [25,26] made a modification (V2) to equation (10) by dividing the process into a finite collection of very small segments of the conversion.

Doing so increases the accuracy of the calculations. This is done by evaluating the Arrhenius integral within small, equidistant $\Delta\alpha$ segments:

$$J\left[E_{\Delta\alpha}, T\left(t_{\Delta\alpha,i}\right)\right] = \int_{t_{\alpha-\Delta\alpha}}^{t_\alpha} \exp\left[-\frac{E_{\Delta\alpha}}{RT\left(t_{\Delta\alpha,i}\right)}\right] \mathrm{d}t \tag{13}$$

or, for a linear heating rate:

$$I\left(E_{\Delta\alpha}, T_{\Delta\alpha,i}\right) = \int_{T_{\alpha-\Delta\alpha}}^{T_\alpha} \exp\left(-\frac{E_{\Delta\alpha}}{RT}\right) \mathrm{d}T \tag{14}$$

This method is known as "advanced Vyazovkin method". Both integrals (13) and (14) can be solved numerically and the solution for (14) can also be found as:

$$I\left(E_{\Delta\alpha}, T_{\Delta\alpha,i}\right) = \frac{E_{\Delta\alpha}}{R} F(T_{\Delta\alpha}) + T_\alpha \exp\left(-\frac{E_{\Delta\alpha}}{RT_\alpha}\right) - T_{\alpha-\Delta\alpha} \exp\left(-\frac{E_{\Delta\alpha}}{RT_{\alpha-\Delta\alpha}}\right) \tag{15}$$

with

$$F(T_{\Delta\alpha}) = E_i\left(-\frac{E_{\Delta\alpha}}{RT_\alpha}\right) - E_i\left(-\frac{E_{\Delta\alpha}}{RT_{\alpha-\Delta\alpha}}\right) \tag{16}$$

where: E_i is the integral exponential defined by: $-E_i(-x) \equiv \int_x^\infty [\exp(-\tau)/\tau]\,\mathrm{d}\tau$ and $\Delta\alpha$ should be kept as small as possible in order to reach accurate computations. The activation energy is that which minimizes equation (10) or equation (11).

3. Design of the module

The module is object oriented with two classes, one to manipulate and organize the experimental data to perform isoconversional computations called `DataExtraction`, and the other to perform the isoconversional computations over the organized data, called `ActivationEnergy`. This module is intended for use in **I**ntegrated **D**evelopment **E**nvironments (IDEs) such as Jupyter Notebook.

The structure of the module was kept simple to make it easy to use based on the usual procedure to obtain activation energy values by means of isoconversional computations. Such procedure consists in reading the raw data obtained from the instrument with a suitable application, with spreadsheets or spreadsheet like interfaces. Selecting a temperature analysis interval to calculate conversion values. Afterwards, group temperature, time or conversion rate values according to the isoconversional criteria. Finally, perform the corresponding computations to obtain the activation energy for each conversion value selected in the analysis.

These functions were implemented with the next dependencies: numpy [27], pandas [28], scipy [29], matplotlib [30] and derivative [31]. A flowchart showing how the functions are organized into two groups is shown in Fig. 2.

4. Workflow

To show the intended use for the module a series of thermograms were simulated. To model thermogravimetric data one can take the Friedman method (equation (5)) and replace the infinitesimal differential with finite differences [32]:

$$\alpha_i = \alpha_{i-1} + \Delta t\left[A_{\alpha_i} f(\alpha_i)\right] \exp\left[-\frac{E_{\alpha_i}}{RT\left(t_{\alpha_i}\right)}\right] \tag{17}$$

where $\Delta t = t_{\alpha_i} - t_{\alpha_{i-1}}$ [33]. The conversion as a function of temperature can be obtained by giving values to: the activation energy, the pre-exponential factor, the reaction model and the temperature program of the equation (17). The transformation to the thermogravimetric curve can be done with the help of the equation (18) in which m_0 and m_f are the initial and final masses, respectively. Equation (18) is derived from (3) replacing ζ by the mass (m).

$$m_i = m_0 - \alpha_i(m_0 - m_\mathrm{f}) \tag{18}$$

The thermograms were simulated using the reaction model: $f(\alpha) = 1-\alpha$; and three cases:

i) **Cnt**: one step with constant $E_\alpha = 75\ \mathrm{kJ\,mol^{-1}}$, with $\ln\left(A_\alpha/\mathrm{min}^{-1}\right) = 12$,

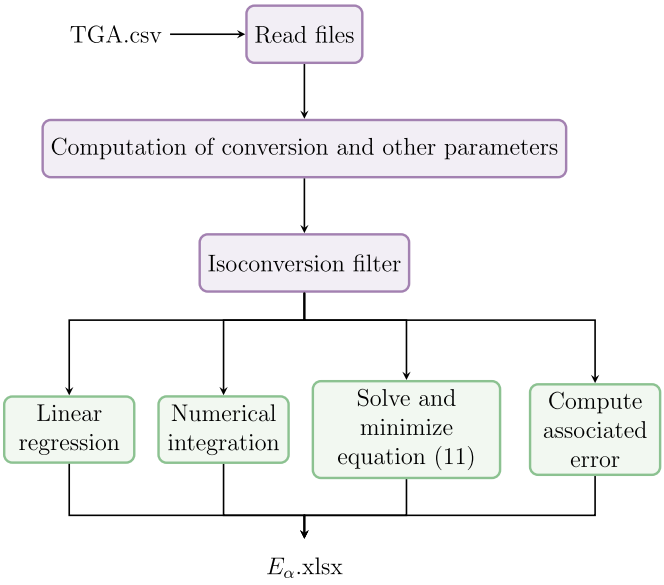


Fig. 2. Flowchart showing the functions that can be performed with the module and how they can be organized into two classes: `DataExtraction` (fuchsia) and `ActivationEnergy` (green). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

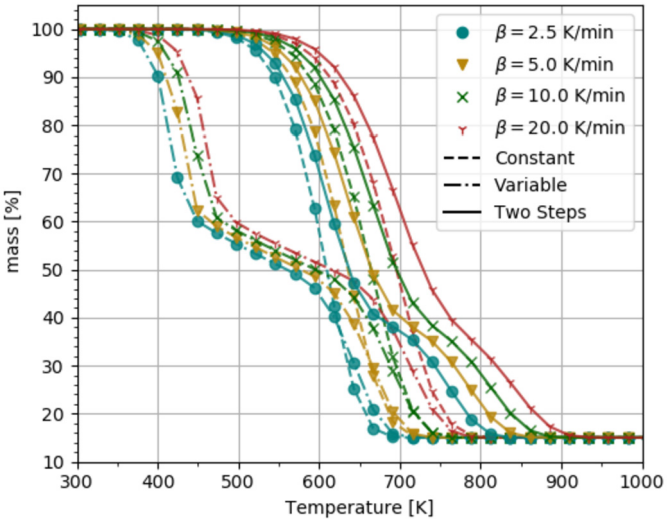


Fig. 3. Thermograms for the three simulated processes. Dashed line: constant activation energy; dash-pointed line: variable activation energy; solid line: two activation energy values.

ii) **Var**: one step with $E_\alpha = 70\ \mathrm{kJ\,mol^{-1}}$ for $\alpha < 0.4$, then E_α varies linearly up to $105\ \mathrm{kJ\,mol^{-1}}$ from $\alpha = 0.4$ to $\alpha < 0.6$, and $E_\alpha = 105\ \mathrm{kJ\,mol^{-1}}$ for $\alpha \geq 0.6$ with $A_\alpha = \ln\left(A_\alpha/\mathrm{min^{-1}}\right) = 17$, and

iii) **2S**: two steps, each with constant $E_\alpha = 75\ \mathrm{kJ\,mol^{-1}}$ and $125\ \mathrm{kJ\,mol^{-1}}$, with $\ln\left(A_\alpha/\mathrm{min^{-1}}\right) = 12$ and $\ln\left(A_\alpha/\mathrm{min^{-1}}\right) = 17$, respectively.

The three sets were computed with equation (17), at heating rates of: $2.5\ \mathrm{K\,min^{-1}}$, $5\ \mathrm{K\,min^{-1}}$, $10\ \mathrm{K\,min^{-1}}$ and $20\ \mathrm{K\,min^{-1}}$. The simulated thermograms are shown in Fig. 3.

4.1. Installation

The module's files can be downloaded from https://github.com/ErickErock/pICNIK or the installation can be done through the `pip` command (`$ pip install picnik`).

4.2. Reading data files and computing conversion values

The raw data, *i.e.*, each comma-separated values file, must have three columns: time, temperature and the property ζ (in this example the property is mass), in that order.

First, it is convenient to create a list containing the path of the files to be used, note that the files have to be input in ascendant heating rate order as shown in Code 1.

```
>>> f_cnt = ['Ecnt2.5.csv','Ecnt5.csv','Ecnt10.csv','Ecnt20.csv']
>>> f_var = ['Evar2.5.csv','Evar5.csv','Evar10.csv','Evar20.csv']
>>> f_2S  = ['E2S2.5.csv','E2S5.csv','E2S10.csv','E2S20.csv']
```

Code 1: A list object containing the paths to the data file is an obligatory parameter for the `read_files()` method. Each name represents a simulated system as follows: f_cnt: set with constant activation energy; f_var: set with variable activation energy; and f_2S: set with two activation energy values.

Next, the module is imported and the `DataExtraction` object is instantiated. The files list to be used is passed as a parameter to the `read_files()` method which also has the optional parameter `encoding` that has the same available options as the `read_csv()` pandas method [28], `utf8` is the default. The function shows a message with the paths used and the computed heating rates by linear regression of temperature *vs.* time of each data set, and returns two arrays `Beta` and `T0`, both will be needed to instantiate the `ActivationEnergy` object. Then, the `Conversion()` method calculates conversion values for a given temperature range, which lower and upper limits, in [K], are passed to the function as parameters `T0` and `Tf`. The function returns a message showing the temperature range and a plot of the thermogram showing the region that was selected within the aforementioned interval.

The `Isoconversion()` method returns at least three isoconversional `DataFrames`: one for temperatures, one for times and another one for conversion rates, that are also stored as the attributes `TempIsoDF`, `timeIsoDF` and `diffIsoDF` respectively. This three isoconversional `DataFrames` are to be used for the Fr, OFW, KAS and V1 methods. The method has four optional parameters: `advanced`, `method`, `N` and `d_a`. The last three are ignored if `advanced` is set to `False`. If it is set to `True`, two more isoconversional `DataFrames` are returned, computed with either of the two available methods: `points` to build an array with a certain number of points which goes as value for `N`; or `step` to create a conversion array with a custom value for $\Delta\alpha$, which goes as value for `d_a`. The advanced isoconversional `DataFrames` are stored as the attributes `TempAdvIsoDF` and `timeAdvIsoDF`, both can be used for the V2 method.

```
>>> import picnik as pnk
>>> xtr = pnk.DataExtraction()
>>> Beta, T0 = xtr.read_files(f_X,encoding='utf8')      #X: cnt, 2S, var
Files to be used:
[EX2.5.csv, EX5.csv, EX10.csv, EX20.csv]
Reading files and creating DataFrames...
The computed heating rates are:
2.50 K/min
5.00 K/min
10.00 K/min
20.00 K/min
>>> xtr.Conversion(323,823)
The temperature range was set to (323.0,823.0) K
Computing conversion values...
Done
>>> TDF,tDF,dDF,TaDF,taDF = xtr.Isoconversion(advanced=True,
                                              method='points',
                                              N = len(xtr.TempIsoDF))
Creating Isoconversion DataFrames...
Done
```

Code 2: The `DataExtraction` object is instantiated. The data is read with the `read_files()` method. The `Conversion()` method sets a temperature range and the `Isoconversion()` method creates the Isoconversion Data Frames.

The isoconversional `DataFrames` are created by taking the data set containing the least points. Taking the values of α of that set as the x axis values for interpolation functions with cubic splines, implemented with scipy, of conversion *vs.* temperature [K], conversion *vs.* time [min] and conversion *vs.* conversion rate for each data set, a `DataFrame` is created where the indexes are the values of α used in the interpolations and each column has the values of the isoconversional temperature, time or conversion rate, for each Heating Rate (HR) and it is named "HR β_i K/min".

The advanced isoconversional `DataFrames` are created apart because of the condition of having a conversion array at equidistant conversion values, and that is not assured for the experimental values. So, a new index is created from an array of conversions made with the numpy functions `linspace()` or `arange()`, depending on which property of the array the user wants to set, the number of points in the array or the magnitude of the interval between conversion ($\Delta\alpha$). The default values are shown in the Code 2.

The data can be visualized with the matplotlib library [30] or with the implemented methods: `get_avsT_plot()` (conversion *vs.* Temperature), `get_avst_plot()` (conversion *vs.* time), `get_dadtvsT_plot()` (conversion rate *vs.* temperature) and `get_dadtvst_plot()` (conversion rate *vs.* time).

Table 1
Isoconversional `DataFrame` associated with the method that utilizes it.

Method	Data frame
`KAS()`, `OFW()`, `Vy()`	`TempIsoDF`
`Fr()`	`diffIsoDF+TempIsoDF`
`aVy()`	`timeAdvIsoDF/TempAdvIsoDF`

4.3. Calculating the activation energies

To create an `ActivationEnergy` object the constructor needs two obligatory parameters (see Code 3). First, `Beta`, is the array of ordered heating rates, which is the attribute `Beta` of `DataExtraction`. The second, `T0`, the array of initial temperatures, also an attribute of `DataExtraction`. Both parameters are also the output of the `read_files()` method. The four optional parameters correspond to isoconversional `DataFrames`, one of temperatures, one of conversion rates, this two for the Fr, OFW, KAS and V1 methods, and the other two are the advanced isoconversional `DataFrames` of temperatures and of times. Although the four latter parameters are optional, they are required for their respective isoconversional method(s) which are as shown in Table 1.

```
>>> ace = pnk.ActivationEnergy(Beta,      #or xtr.Beta
                               T0,        #or xtr.T0
                               TDF,       #or xtr.TempIsoDF
                               dDF,       #or xtr.diffIsoDF
                               TaDF,      #or xtr.TempAdvIsoDF
                               taDF)      #or xtr.timeAdvIsoDF
```

Code 3: Creating an `ActivationEnergy` instance.

The activation energy can be calculated with five isoconversional methods: `Fr()`, `OFW()`, `KAS()`, `Vy()` and `aVy()`. All five methods return a tuple containing an array of activation energy values as first element and as second, an array of the associated error, both in $\mathrm{kJ\,mol^{-1}}$. Only the `Fr()` method returns a tuple with three elements, third one being an array of $\ln[A_\alpha f(\alpha)]$ values which can be used to make isoconversional predictions [32,33].

```
>>> E_Fr  = ace.Fr()
Friedmann method: Computing activation energies...
Done
>>> E_OFW = ace.OFW()
Ozawa-Flynn-Wall method: Computing activation energies...
Done
>>> E_KAS = ace.KAS()
Kissinger-Akahira-Sunose method: Computing activation energies...
Done
>>> E_Vy = ace.Vy(bounds=(1,300),
                  method='senum-yang')
Vyazovkin method: Computing activation energies...
Done
>>> E_aVy = ace.aVy(bounds = (1,300),
                    var = 'time',
                    method ='trapezoid')
Advanced Vyazovkin method: Computing activation energies...
Done
```

Code 4: Computing activation energies with the five isoconversional methods.

The first three methods do not need a parameter. The `Vy()` method takes two optional parameters: a tuple of two elements being the lower and upper limits for the interval of values of E_α in $\mathrm{kJ\,mol^{-1}}$ for which equation (11) will be evaluated in search of a minimum. To visualize if the default bounds are pertinent for the minimization, the `visualize_omega()` method is helpful. The next parameter is `method`, for solving integral (7), which accepts four options: `senum-yang` for the Senum-Yang approximation, `trapezoid` for the trapezoid rule of numerical integration, `simpson` for the simpson rule and `quad` for using a technique from the Fortran library QUADPACK implemented in the `scipy.integrate` subpackage. The default parameters are shown in Code 4. The associated error is computed by the procedure described by Vyazovkin and Wight [34], and implemented in the `error_Vy()` method.

The `visualize_omega()` method takes an obligatory parameter which is the row of the isoconversional `DataFrame`, *i.e.*, the implicit index of the conversion value in the Temperature `DataFrame`, for which omega will be evaluated; and three optional arguments: `bounds`, having the interval (1,300) as default, `N`(=1000) which is the number of points in-between the bounds and `method` which is the same parameter as in the `Vy()` method; and returns a matplotlib plot of E_α *vs.* $\Omega(E_\alpha)$, showing the value of α corresponding to the index (row). An example of this method is shown in Fig. 4 for the simulated systems, and the code is shown in Code 5.

The `aVy()` method receives three optional parameters: `bounds` with the same interpretation as in `Vy()` (and there is also a `visualize_advomega()` method), `var` which indicates over which variable the integral is going to be computed, the options are

```
>>> ace.visualize_omega(row,
                        bounds,
                        N = 1000,
                        method='senum-yang')
```

Code 5: Method to visualize equation (11).

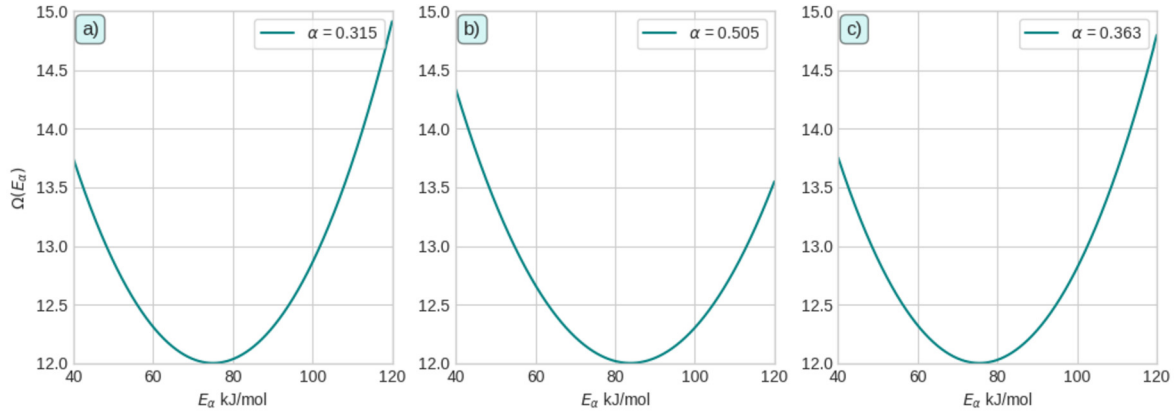


Fig. 4. Bounds evaluation for $\Omega(E_\alpha)$; (a) E_α constant, row = 690; (b) variable E_α, row=420 and; (c) two E_α values, row = 748.

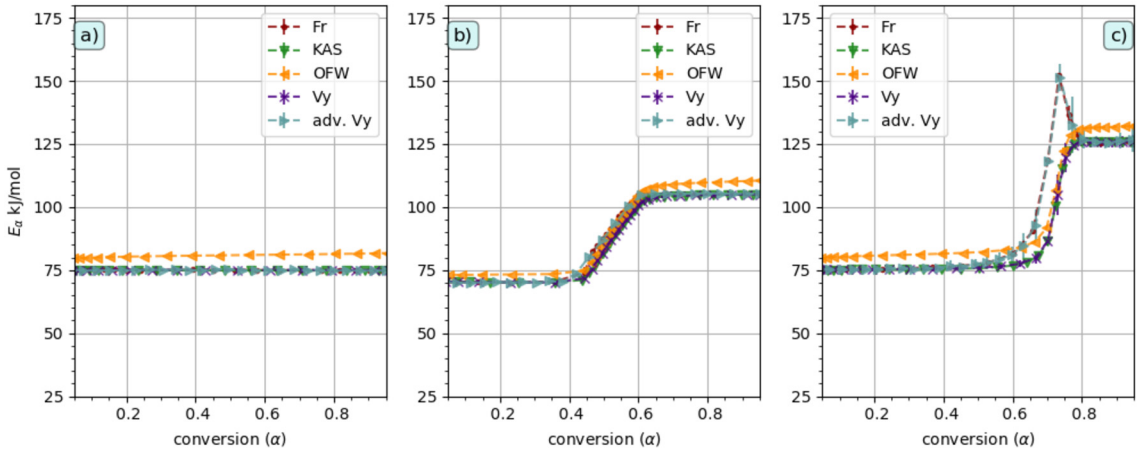


Fig. 5. Activation energies obtained with the five implemented isoconversional methods for the simulated processes: a) Constant activation energy ($E_\alpha = 75$ kJ mol^{-1}), b) Variable activation energy ($E_\alpha = 70$ kJ mol^{-1} for $\alpha < 0.4$, then E_α varies linearly up to 105 kJ mol^{-1} from $\alpha = 0.4$ to $\alpha < 0.6$, and $E_\alpha = 105$ kJ mol^{-1} for $\alpha \geq 0.6$) and c) Two steps ($E_\alpha = 75$ kJ mol^{-1} and 125 kJ mol^{-1}).

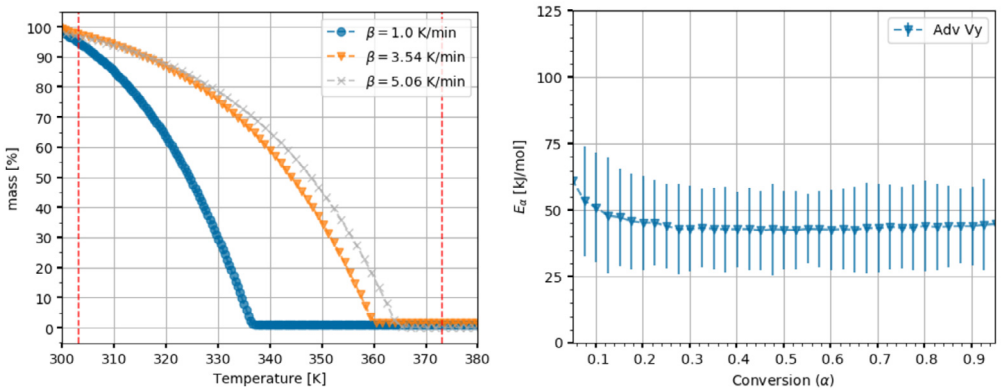
Table 2
Activation energies, associated errors and relative errors obtained for the simulated system with $E_\alpha = 75$ kJ mol^{-1}, in a conversion interval of 0.05 to 0.95.

Method	$\langle E_\alpha \rangle$ [kJ mol^{-1}]	$\langle e \rangle$ [kJ mol^{-1}]	e_{rel} [%]
Fr	75.01	0.06	0.01
OFW	80.7	0.1	7.6
KAS	74.87	0.01	0.17
Vy	75.02	0.02	0.03
aVy	75.0	0.5	0

`Temperature` and `time`. If the option `Temperature` is given, the integral will be computed as equation (15), and if the `time` option is selected, another parameter is to be specified: `method`, for a numerical integration method with the available options: `trapezoid`, `simpson` and `quad`. The default options are shown in Code 4. The associated error is computed the same way as for the `Vy()` method, but with the appropriate integral implemented in `error_aVy()`. Fig. 5 shows that the activation energies obtained for the three simulated systems are in perfect agreement with the programmed values, except for the `Fr()` and `aVy()` methods in the conversion value for the step transition of the two steps system. A comparison of the values obtained for the constant energy system with the programmed value is shown in Table 2.

```
>>> ace.export_Ea(E_Fr  = True,
                  E_OFW = True,
                  E_KAS = True,
                  E_Vy  = True,
                  E_aVy = True,
                  file_t= "xlsx" )
Exporting activation energies...
Results saved as Activation_Energies_Results.xlsx and
Advanced_Vyazovkin_Results.xlsx
Done.
```

Code 6: Exporting results with the `export_Ea method`.



(a) (b)

Fig. 6. Vaporization of n-decane. a) Thermogram showing the analysis range, output of the `Conversion()` method. b) Activation energy profile obtained with the `aVy()` (V2) method.

4.4. Exporting results

To export the results, the `export_Ea()` method returns a `xlsx` spreadsheet or a `csv` file with the conversion values and average isoconversional temperature along with a pair of column per method with the E_α values and its error. The parameters of the method are `E_Fr`, `E_OFW`, `E_KAS`, `E_Vy` and `E_aVy`, with optional boolean values. `False` is the default, if the method is called like that, the output file will have only the first two columns. If a parameter or more are set to `True` the corresponding pair of columns are added to the output file. The only parameter that returns another file of results is `E_aVy`, which has different lengths for the conversion and mean temperatures arrays. The last parameter, `file_t`, indicates the type of file to be created, the optional values are `csv` and `xlsx`.

5. Case studies

To validate the code, the vaporization of n-decane and the thermal degradation of polypropylene (PP) were studied, and the results obtained with pICNIK were compared with those reported in the literature.

5.1. Vaporization of n-decane

According to Ekawa et al. [35] the activation energy should be very similar to the vaporization enthalpy ($\Delta_v H$). The vaporization enthalpy of the n-decane has a mean value of 47.4 kJ mol^{-1} [36].

The n-decane was purchased from Sigma Aldrich, with purity ≥99% and used without further purification. TGA experiments were carried out with a SDT Q600 from TA instruments. The heating programs used were: 1.0 K min^{-1}, 3.5 K min^{-1} and 5.0 K min^{-1}. Initially 12(1) mg of n-decane were placed on ceramic pans (90 µL). All measurements were carried out under N_2 atmosphere with a total flow rate of 100 mL min^{-1}, from room temperature to 523 K. Code 7 shows the commands needed to perform the data treatment with the pICNIK module. In less than 40 lines of code the activation energy as a function of conversion and temperature is computed.

Ekawa et al. [35] reported activation energy of 52(1) kJ mol^{-1} applying the V2 method. Their experimental conditions were: heating programs of 1.0 K min^{-1}, 1.5 K min^{-1}, 2.3 K min^{-1}, 3.5 K min^{-1} and 5.0 K min^{-1}; initial mass of 17.5 mg; and N_2 flow rate of 80 mL min^{-1}.

Fig. 6a shows the mass loss (n-decane liquid-vapor evaporation process) between room temperature and 373 K. In general terms, this result describes the dependence relationship between saturation pressure (P_s) and temperature, and reveals the influence of heating rate on the limit temperature where phase boundary liquid-vapor vanishes [35]. From the kinetic point of view, the dependence E_α *vs.* α, describes the typical expected behavior of a pure liquid, where the activation energy is independent of the conversion and the evaporation rate depends only on the temperature, *i.e.*, a process of zero order [37].

Activation energy as function of conversion is shown in Fig. 6b. A constant behavior is observed with mean value of 46(17) kJ mol^{-1}. Based on this activation energy, a vaporization enthalpy of around 49 kJ mol^{-1} can be estimated, which is between the values reported by NIST (47.4 kJ mol^{-1} [36]) and Ekawa (55 kJ mol^{-1} [35]). These results show that pICNIK provides results in agreement with those reported.

```
>>> import picnik as pnk.
>>> Dfiles = ["DEC 1 N2.csv",
                "DEC 3.5 N2.csv",
                "DEC 5 N2.csv"]
>>> xtr = pnk.DataExtraction()
>>> DB, DT0 = Dxt.read_files(Dfiles,encoding='utf16')
Files to be used:
[Decano/DEC_1.csv, Decano/DEC_3.5.csv, Decano/DEC_5.csv]
Reading files and creating DataFrames...
The computed heating rates are:
1.00 K/min
3.54 K/min
5.06 K/min
>>> xtr.Conversion(303,373)
The temperature range was set to (303.0,373.0) K
Computing conversion values...
Done
>>> DT, Dt, Dd, DaT, Dat = Dxt.Isoconversion(advanced=True)
Creating Isoconversion DataFrames...
Done
>>>ace = pnk.ActivationEnergy(DB,            #or xtr.Beta
                              DT0,           #or xtr.T0
                              DT,            #or xtr.TempIsoDF
                              Dd,            #or xtr.diffIsoDF
                              DaT,           #or xtr.TempAdvIsoDF
                              Dat)           #or xtr.timeAdvIsoDF

>>> E_aVy = ace.aVy((20,210))
Advanced Vyazovkin method: Computing activation energies...
Done.
>>> Dace.export_Ea(E_aVy=True)
Exporting activation energies...
Results saved as Advanced_Vyazovkin_Results.xlsx
Done.
```

Code 7: Usage of pICNIK to compute the activation energy for the vaporization of n-decane.

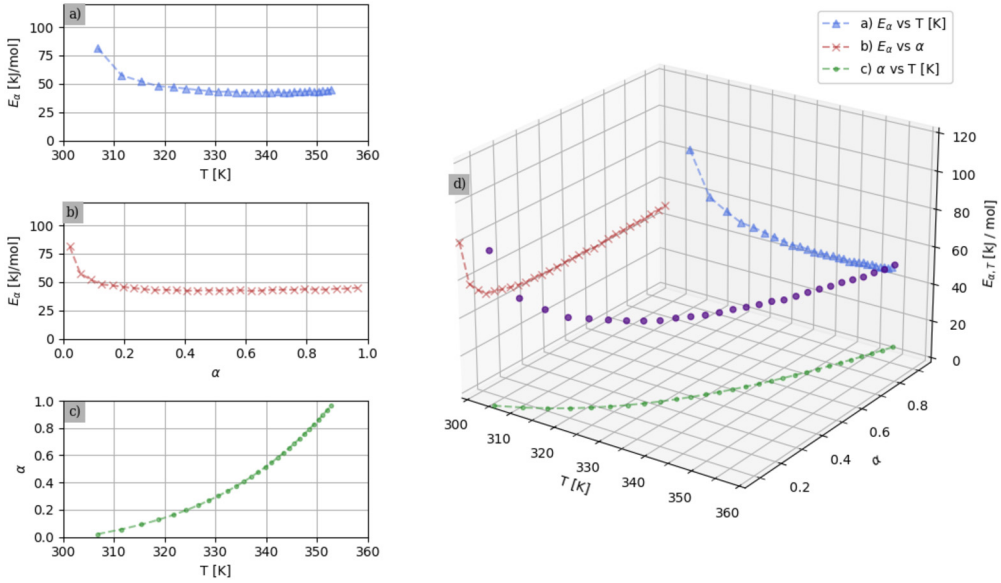


Fig. 7. Summary of the kinetic process of the vaporization of n-Decane computed with the V2 method. a) Activation energy (E_α) in kJ mol^{-1} *vs.* temperature in K; b) Activation energy (E_α) in kJ mol^{-1} *vs.* conversion (α); c) Conversion (α) profile as a function of temperature in K and; d) Global relationship between $E_{\alpha,T}$, α and T.

As summary, Fig. 7 shows the three main profiles of the thermokinetic process as well as the global behavior for the vaporization of n-Decane, computed with the V2 method.

```
>>> import picnik as pnk.
>>> PPf = ["PP_1.csv",
           "PP_ 2_3.csv",
           "PP_ 5_B3.csv",
           "PP_20_B3.csv"]
>>> PPxt = pnk.DataExtraction()
>>> PPxt.read_files(PPf,encoding='utf8')
Files to be used:
[PP/Batch_3/PP_1_B3.csv, PP/Batch_3/PP_ 2_3_B3.csv,
PP/Batch_3/PP_5_B3.csv, PP/Batch_3/PP_20_B3.csv]
Reading files and creating DataFrames...
The computed heating rates are:
1.00 K/min
2.31 K/min
5.02 K/min
20.10 K/min
>>> PPxt.Conversion(473,773)
The temperature range was set to (473.0,773.0) K
Computing conversion values...
Done
>>> xtr.Isoconversion()
Creating Isoconversion DataFrames...
Done
>>> PPace = pnk.ActivationEnergy(PPxt.Beta,
                                 PPxt.T0,
                                 PPxt.TempIsoDF,
                                 PPxt.diffIsoDF)
>>> PPKAS = PPace.KAS()
>>> PPFr = PPace.Fr()
>>> PPace.export_Ea(E_Fr=True, E_KAS=True)
Exporting activation energies...
Results saved as Activation_Energies_Results.xlsx
Done.
```

Code 8: Data treatment with pICNIK for PP's thermal degradation.

5.2. Thermal degradation of polypropylene

Based on previous studies, PP thermal degradation is reported to have a conversion-dependent activation energy, with a gradual increase in the activation energy values as conversion increases. Gao et al. studied the degradation of isotactic PP with the KAS method and obtained an increasing activation energy between 115 $\mathrm{kJ\,mol^{-1}}$ and 140 $\mathrm{kJ\,mol^{-1}}$ ascribing those results to the random chain-scission and deviatons from the stationary state [2]. Lecouvet et al. used the Fr method to obtain an activation energy that increases from 105 $\mathrm{kJ\,mol^{-1}}$ to 150 $\mathrm{kJ\,mol^{-1}}$ and propose that this behavior is the effect of two consecutive processes, random chain scission followed by all the degradation reactions: end-chain and mid-chain β-scission, hydrogen transfer and hydrogen abstraction [3].

The PP was purchased from Sigma Aldrich as pellets of isotactic polypropylene with average molecular weight of 12000 $\mathrm{kg\,mol^{-1}}$. Each sample was shaped into disks of masses of about 10(1) mg. TGA measurements were carried out with a SDT Q600 from TA instruments. The heating programs were: 1.0 $\mathrm{K\,min^{-1}}$, 2.3 $\mathrm{K\,min^{-1}}$, 5 $\mathrm{K\,min^{-1}}$ and 20 $\mathrm{K\,min^{-1}}$. All thermogravimetric experiments were carried out under N_2 atmosphere, with a total flow rate of 100 $\mathrm{mL\,min^{-1}}$ and from room temperature to 873 K.

In order to compare the activation energy obtained with pICNIK, the methods of KAS and Fr were used as shown in Code 8 so it was not necessary to set the `advanced` parameter to `True` and neither was the parameters `TempAdvIsoDF` and `timeAdvIsoDF` into the `ActivationEnergy` instance.

According to the results of this study (see Fig. 8b), the slight increase from 124 $\mathrm{kJ\,mol^{-1}}$ to 132 $\mathrm{kJ\,mol^{-1}}$ does not represent a substantive difference to describe this thermokinetic behavior in mechanistic terms. This small difference $\left(\text{less than } 10\ \mathrm{kJ\,mol^{-1}}\right)$ with respect to that reported by others [2,3], indicates that under the experimental conditions of this study, the rate-controlling step is transport phenomena, which largely depends on physical properties (density, thermal conductivity and heat capacity) of the sample. Fig. 9 shows a summary of the thermokinetic behavior of PP through the dependence relationship of the activation energy as a function of conversion and temperature.

6. Conclusions

A Python module with methods to compute activation energy of thermally stimulated processes has been designed, developed and validated. This module will reduce the time involved in kinetic parameter calculations and thus allow faster decision making. The module is registered in the python package index (PyPI) under an MIT license. The code is functional and has been validated by studying the evaporation of n-decane, the thermal degradation of polypropylene and a series of simulated processes. The permissive free software license combined with its object-oriented programming, provide the flexibility to integrate the code into other projects, easily modify it for specific applications and further develop it by the user community.

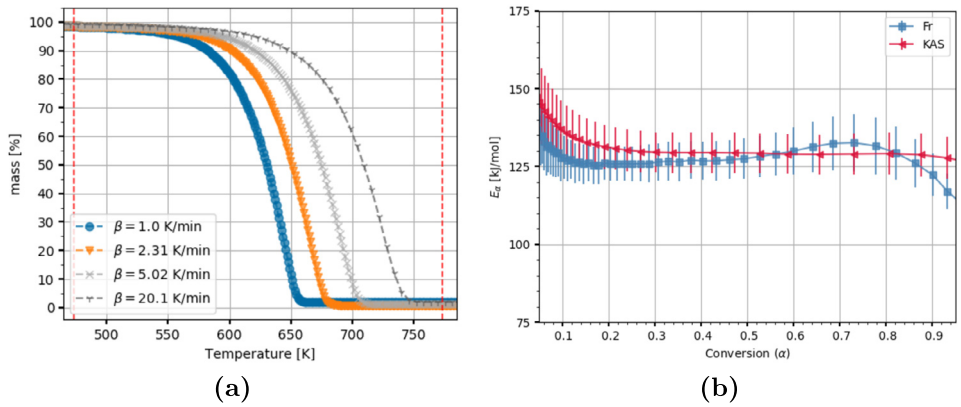


Fig. 8. Thermal degradation of `PP`. a) Thermogram showing the analysis range, output of the `Conversion()` method. b) Activation energy profile obtained with the `Fr()` and `KAS()` methods.

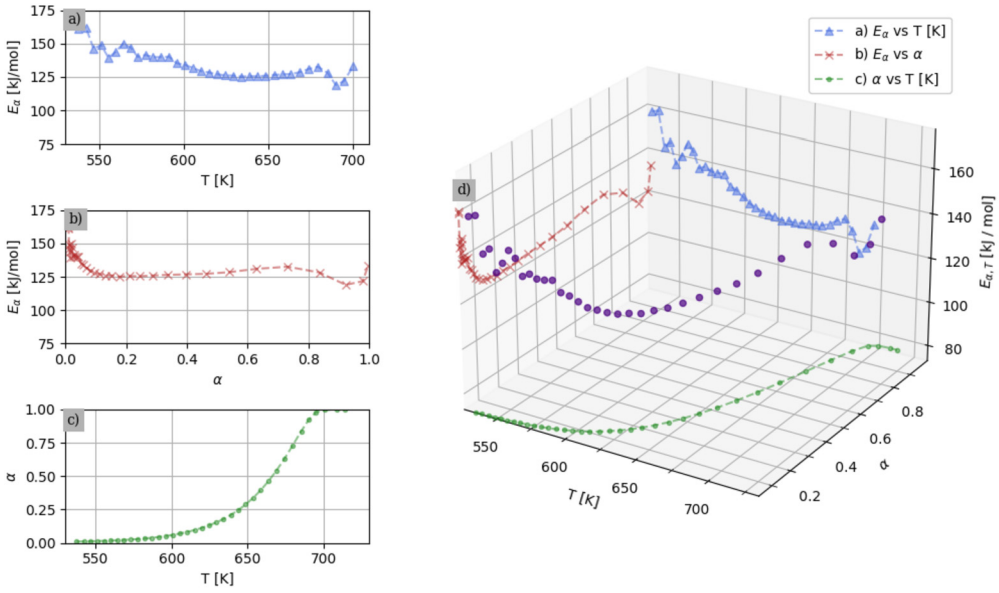


Fig. 9. Summary of the kinetic process of the thermal degradation of PP computed with the Fr method. a) Activation energy (E_α) in kJ mol^{-1} *vs.* temperature in K; b) Activation energy (E_α) in kJ mol^{-1} *vs.* conversion (α); c) Conversion (α) profile as a function of temperature in K and; d) Global relationship between $E_{\alpha,T}$, α and T.

The functions that can be performed with this module are:

1. Read files containing data from multiple TGA experiments.
2. Given a temperature range to analyze, compute the conversion according to equation (3).
3. Compute the conversion rate with numerical differentiation.
4. Apply an isoconversional filter to align temperatures, times and conversion rates.
5. Five methods to perform isoconversional computations: Friedman, Ozawa-Flynn-Wall, Kissinger-Akahira-Sunose and Vyazovkin's methods for constant and variable activation energies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors thank the financial support provided by DGAPA-UNAM through PAPIIT project IG100618.

References

[1] M.E. Brown, et al., Thermochim. Acta 355 (1–2) (2000) 125–143.
[2] Zhiming Gao, et al., Polym. Degrad. Stab. 80 (2) (2003) 269–274.
[3] Benoît Lecouvet, et al., Polym. Degrad. Stab. 97 (9) (2012) 1745–1754.
[4] Paola Brachi, Victor Santes, Enelio Torres-Garcia, Fuel 302 (2021) 120995.

[5] Enelio Torres-Garcia, Paola Brachi, J. Therm. Anal. Calorim. 139 (2) (2020) 1463–1478.
[6] E. Torres-García, L.F. Ramírez-Verduzco, J. Aburto, Waste Manag. 106 (2020) 203–212.
[7] J.A. Caballero, et al., J. Anal. Appl. Pyrolysis 42 (2) (1997) 159–175.
[8] Maider Amutio, et al., Fuel 95 (2012) 305–311.
[9] Xu Gao, Lin Jiang, Qiang Xu, J. Hazard. Mater. 386 (2020) 121645.
[10] Martina Maria Calvino, et al., Thermochim. Acta 700 (2021) 178940.
[11] Dmitry Drozin, et al., SoftwareX 11 (2020) 100359.
[12] Python Software Foundation. The Python Package Index (Accessed 2022-02-28)..
[13] M. Brown, J. Therm. Anal. Calorim. 49 (1) (1997) 17–32.
[14] Sergey Vyazovkin, et al., Thermochim. Acta 520 (1) (2011) 1–19.
[15] Sergey Vyazovkin, et al., Thermochim. Acta 590 (2014) 1–23.
[16] Sergey Vyazovkin, Charles A. Wight, Int. Rev. Phys. Chem. 17 (3) (1998) 407–433.
[17] Henry L. Friedman, J. Polym. Sci., C Polym. Symp. 6 (1) (1964) 183–195.
[18] C.D. Doyle, J. Appl. Polym. Sci. 5 (15) (1961) 285–292.
[19] Takeo Ozawa, Bull. Chem. Soc. Jpn. 38 (11) (1965) 1881–1886.
[20] Joseph H. Flynn, Leo A. Wall, J. Polym. Sci., Part B, Polym. Lett. 4 (5) (1966) 323–328.
[21] Homer E. Kissinger, Anal. Chem. 29 (11) (1957) 1702–1706.
[22] A.W. Coats, J.P. Redfern, Nature 201 (4914) (1964) 68–69.
[23] Sergey Vyazovkin, David Dollimore, J. Chem. Inf. Comput. Sci. 36 (1) (1996) 42–45.
[24] G.I. Senum, R.T. Yang, J. Therm. Anal. 11 (3) (1977) 445–447.
[25] Sergey Vyazovkin, J. Comput. Chem. 18 (3) (1997) 393–402.
[26] S. Vyazovkin, J. Comput. Chem. 22 (2) (2001) 178–183.
[27] Numpy project. Numpy documentation. Version 1.19.1 (Accessed 2021-06-25)..
[28] Pandas documentation. Version 1.2.2 (Accessed 2021-06-28)..
[29] Scipy documentation. Version 1.6.3 (Accessed 2021-06-26)..
[30] Python Software Foundation. Matplotlib documentation. Version 3.1.2 (Accessed 2021-06-30)..
[31] Derivative documentation (Accessed 2021-06-20)..
[32] J. Farjas, P. Roura, J. Therm. Anal. Calorim. 109 (1) (2012) 183–191.
[33] Lèrys Granado, Nicolas Sbirrazzuoli, Thermochim. Acta 697 (2021) 178859.
[34] Sergey Vyazovkin, Charles A. Wight, Anal. Chem. 72 (14) (2000) 3171–3175.
[35] Bruno Ekawa, Victoria L. Stanford, Sergey Vyazovkin, J. Mol. Liq. 327 (2021) 114824.
[36] NIST Chemistry WebBook, Data obtained from the mean of several values for the experimental temperature range (Accessed 2022-02-26)..
[37] Sergey Vyazovkin, Int. Rev. Phys. Chem. 39 (1) (2020) 35–66.