# Modeling of interstitials diffusion during debinding/sintering of 3D printed metallic filaments: Application to titanium alloy and its embrittlement

M. Coffigniez, Laurent Gremillard, M. Perez, S. Simon, C. Rigollet, Erik Bonjour, Patrick Jame, X. Boulnat

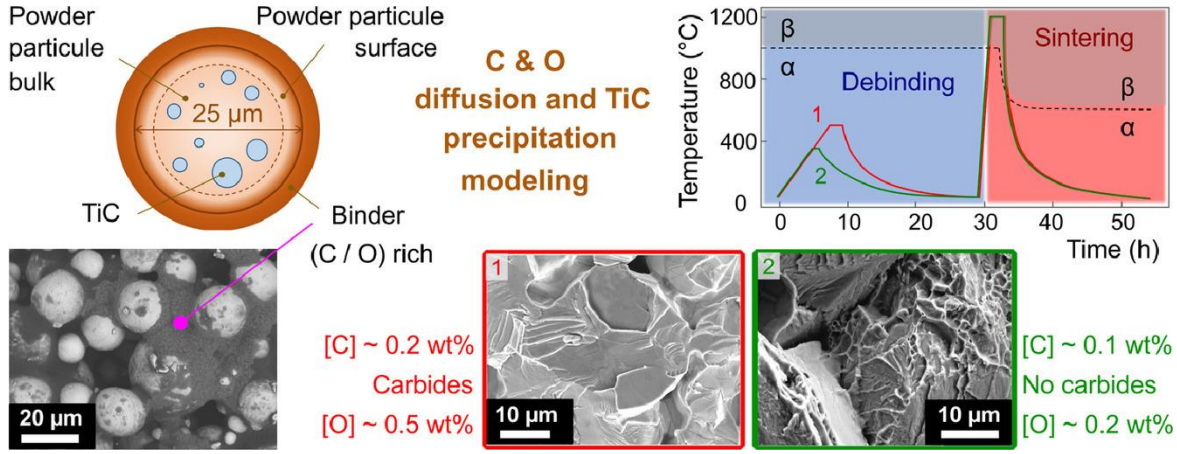
# Modeling of interstitials diffusion during debinding/sintering of 3D printed metallic filaments: Application to titanium alloy and its embrittlement

Published in Acta Materialia 219 n° 117224 (2021)
https://doi.org/10.1016/j.actamat.2021.117224

M.Coffigniez[a], L.Gremillard[a], M.Perez[a], S.Simon[b], C.Rigollet[b], E.Bonjour[c], P.Jame[c], X.Boulnat[a*]

[a] Université de Lyon, INSA-Lyon, CNRS, MATEIS UMR5510, 7 Av. Jean Capelle, F-69621 Villeurbanne Cedex, France.
[b] Université de Lyon, ECAM Lyon, LabECAM, 40 montée Saint Barthélémy, Lyon F-69005, France
[c] Institut des Sciences Analytiques, UMR CNRS 5280, Villeurbanne, France

* Corresponding author: Xavier.boulnat@insa-lyon.fr




## Keywords
Titanium alloys; Diffusion; Oxygen Embrittlement; Additive Manufacturing

## Abstract
Parts made by metal additive manufacturing processes assisted by sintering suffer from embrittlement due to binder/powders interactions. Using a core-shell approach, we modelled the diffusion of carbon and oxygen from the binder into the metallic powders. The model was assessed by measuring the interstitials content upon various debinding/sintering steps. The diffusion/precipitation phenomena during these non-isothermal treatments lead to either a saturated solid solution or the precipitation of titanium carbides. The consequences on the mechanical behavior of single 3D printed filaments were quantified by bending tests, highlighting a transition from brittle to ductile fracture depending on the debinding parameters. This approach can be applied to understand the role of fast diffusing interstitial elements into various powders systems in order to optimise the chemical composition and the mechanical properties of 3D printed metallic parts.

## 1. Introduction

Some additive manufacturing technologies, such as fused-deposition modeling, direct-ink writing or binder jetting, involve thermal treatments, namely debinding/sintering, after 3D printing. One of the most studied metals using these techniques is Ti-6Al-4V (Ti64) because of its performance for biomedical [1] and aerospace applications [2].

Direct-ink writing (DIW) enables to build-up structures layer by layer, without fusion, from a powder based ink or paste presenting the adequate rheology [3-5]. Once printed, the part undergoes thermal treatments (drying, debinding, sintering) to reach its final structure and mechanical properties. As shown in a previous work, this "low cost" process enables to produce Ti64 parts presenting bimodal interconnected porosity, which can be interesting for tissue engineering applications [6].

Similarly to metal injection moulding (MIM), 3D printing techniques such as DIW imply the use of a carrier fluid potentially containing several organic components (binder, dispersant, rheology modifiers...) [7-9]. A debinding step is thus necessary to eliminate all traces of organics before final consolidation by sintering [8-10].

In the case of the MIM process, debinding is usually performed in two steps, with a primary debinding that can be performed using solvent or catalytic reaction, and a secondary thermal debinding [11]. Removing the primary binder leads to the formation of an interconnected porosity network that facilitate the escape of the backbone binder during the secondary thermal debinding. Minimising the proportion of the backbone binder can thus be a way to limit the diffusion of atoms contained in this binder into the powder particles [7], [12].

However, as DIW is relying on rheological phenomena that have nothing to do with those involved in MIM, the binders used are also totally different. Parts obtained by DIW are therefore usually debinded through one thermal treatment [8-10]. Thus carbon, oxygen and hydrogen from the binder may penetrate into the metal and therefore affect its final properties. Indeed, Ti64 mechanical properties, mainly tensile strength, elongation, and fracture toughness, highly depend on its interstitial elements content [13], [14]. The debinding temperature is thus a key parameter: a too high temperature would lead to an important C/O/H intake, whereas a too low temperature would not permit the complete elimination of organic phase.

In their pioneering contributions, Li et al. [15, 16] printed Ti64 scaffolds by DIW. Their debinding treatment was performed at 500 °C under high vacuum. Later, Elsayed et al. [17,18] debinded Ti64 scaffolds under the same conditions (500 °C, high vacuum) and showed the presence of carbides in their as-sintered scaffolds. These carbides were suspected to be responsible of the brittle nature of the scaffolds fracture under compression [18]. In addition, in these different works, the sintering treatments were carried out under high vacuum [15-18]. As titanium is easily subject to oxidation, it should be sintered in an oxygen-free atmosphere [7], [19]. Although argon is widely used for practical reasons, vacuum sintering is to be preferred since it allows the removal of volatile pollutants like hydrogen, which might be present as it is one of the binder component. Indeed, it has been reported than hydrogen can be removed above 500 °C under vacuum [7,20,21], which is interesting as hydrogen is also known to lead to titanium and titanium alloys embrittlement [22].

In order to better understand the link between Ti64 embrittlement, C/O intake and carbide precipitation, a coupled modelling approach would be necessary. To the authors knowledge, no such approach is available in the literature. Precipitation models alone are indeed quite common. The coupling with an evolving diffusion profile from the outside to dynamically modify the driving force of precipitation is, on the other hand, rarely encountered. Such coupling has already been proposed in steels [23,24] and aluminium [25], but never in Ti64 to model the precipitation during debinding/sintering processes.

In this paper, we evaluate the effect of the debinding conditions (temperature, holding time, heating rate, atmosphere) on both microstructural and mechanical properties of Ti64 scaffolds obtained by

DIW. To do so we printed our samples thanks to Pluronic F-127 hydrogel which is a commonly used carrier fluid for DIW [6,8,10,26-29]. Then we applied different debinding conditions to our samples before sintering all of them with the same set of conditions, to study the impact of the debinding conditions only.
The C and O intake was systematically measured at each stage of the process (initial powder, as-debinded and as-sintered). In parallel, the microstructure was investigated by X-Ray diffraction and scanning electron microscopy. A diffusion/precipitation model was applied to our process in order to better understand both C and O intake as well as precipitation of titanium carbides during the process. Finally, the resulting mechanical properties are presented and discussed for the different debinding conditions.

## 2. Materials and experimental methods

### 2.1. Fabrication strategy

#### *2.1.1. Ink formulation*

Ti-6Al-4V (Ti64) argon atomised powder has been purchased from TLS Technik GmbH (Germany). Its chemical composition is given in Table 1.

*Table 1. Composition of the Ti64 powder according to the supplier.*

| Chemical element | N | C | H | Fe | O | Al | V | Ti |
|---|---|---|---|---|---|---|---|---|
| **Composition (wt%)** | 0.005 | 0.01 | <0.001 | 0.22 | 0.08 | 6.20 | 3.98 | Bal. |

Pluronic F127 (poly(ethylene oxide)-poly(propylene oxide)-poly(ethylene oxide) triblock co-polymer surfactant, Sigma Aldrich, referred to as "Pluronic F-127" in the following) was used in the form of dried granules for the binder preparation. In solution in water, pluronic F127 forms a thermoreversible gel. As shown in a previous work [6], a 25 wt% Pluronic F-127 hydrogel (gel temperature around 20 °C), loaded with 50 vol% of Ti64 powder represents a good compromise to print Ti64. The Pluronic F-127 was gradually dissolved in distilled water and the obtained solution was stored at 4 °C, to be kept in its liquid state before adding Ti64 particles.
Ti64 addition was carried out in three stages. The batches were mixed in a dual asymmetric centrifugal mixer (Speed-Mixer DAC 150.1 FVZ-K) for 1 min at 1500, 1700 and 2000 rpm after the first, the second and the last addition, respectively. Before each mixing step, inks were kept for 10 min at 4 °C, in order to be mixed when liquid.

#### *2.1.2. 3D printing by DIW*

Printing was performed using a robocaster (3D Inks, LLC, Stillwater, OK, USA) controlled by the Aerotech A3200 motion software (Aerotech Inc., Pittsburgh, PA, USA). Samples were printed on glass plate, covered with a thin layer of grease, to help the removal of dried samples. Inks were extruded at 10 $mm.s^{-1}$ through conical nozzles with internal diameters of 840 µm. For all scaffolds, the rod spacing (distance between two rod centres) was fixed at 1260 µm. An interlayer distance of 672 µm was used to ensure good cohesion between the 7 printed layers. Single filaments were also printed to perform bending test. To ensure a stabilised flow during printing, a sacrificial length (few mm) of ink was extruded before printing each sample. For all samples, the temperature was maintained at 25 °C in the printing chamber, ensuring that inks were printed in their gel state.

#### *2.1.3. Debinding and sintering*

Printed scaffolds were dried 48h at room temperature (above 22 °C) before being removed from the glass substrate. Then, they were thermally debinded and sintered in two separate stages, using a Naberthermn N 11/hr furnace with an in-house set up enabling a controlled atmosphere. Different debinding conditions were studied by combining the following parameters:
- debinding temperature (350 and 500 °C),
- holding time (30 min and 2 h),

- heating rate (1 °C·min$^{-1}$ and 5 °C·min$^{-1}$),
- atmosphere (primary dynamic vacuum (around 5·10$^{-2}$ mbar) or 0.2 L·min$^{-1}$ argon flow).

Finally, samples were sintered 2h at 1200 °C under dynamic secondary vacuum (below 5·10$^{-4}$ mbar), using a heating rate of 10 °C·min$^{-1}$ and the natural cooling rate of the furnace that decreases from 50 °C·min$^{-1}$ to less than 1 °C·min$^{-1}$ as the temperature drops.

## 2.2. Characterisation

### *2.2.1. Thermogravimetric analysis*

Thermogravimetric analysis was conducted using a TGA Q50 analyser, equipped with an EGA furnace (TA Instruments, New Castle, USA). Mass losses were recorded during a heating rate of 1 °C·min$^{-1}$ from room temperature to 500 °C under of 150mL·min$^{-1}$ nitrogen flow (after a purging step).

### *2.2.2. Elementary analysis*

Carbon content measurements were carried out using an Inductar CS cube analyser from Elementar Analysensysteme GmbH (Langenselbold, Germany). Samples were weighed into ceramic crucibles with W/Sn and Fe as combustion accelerators. The equipment performed sample combustion under oxygen flow using an induction furnace. The carbon extracted from the sample was transformed into carbon dioxide, quantified by $CO_2$ non dispersive infrared (NDIR) detector. Measurements were performed in three replicates for each sample. TiC powder was also used as unknown sample to verify the system ability to analyse TiC. In addition, 58A SY13001-1 and 173c titanium based standard reference materials (HRT Labortechnik GmbH Buchholz, Germany) were used to ensure a lack of drift on the analyser calibration.

Oxygen contents were obtained using an EMGA 620 W analyser from Horiba Jobin Yvon company. Samples were weighed into nickel capsules (Lüdiswiss, Flawil, Switzerland) and three measurements were performed for each sample. The system used high-temperature fusion in a helium (He) impulse furnace to extract oxygen. Nickel capsules containing samples drop into a graphite crucible raised to high temperature (between 2500 and 2800 °C) in a flow of helium. The oxygen which was transformed into carbon monoxide (CO) by combining with the carbon in the crucible, was quantified by a non-dispersive infrared CO cell. Then the carbon monoxide was oxidised into carbon dioxide and trapped. The 173c standard reference titanium materials was also used to verify the analyser calibration and the robustness of the pyrolysis reaction. Before the experiments, a linear calibration was established by passing several standards. In addition some samples were sent to the Elementar Application Laboratory to obtain hydrogen and nitrogen content. Samples were weighed into nickel capsules and analysed using an inductar EL cube in ONH mode (Elementar Analysensysteme GmbH, Germany). Four measurements were performed on each batch.

All of these measures provide a complete overview of binder induced pollution, which includes carbon, oxygen and hydrogen, but also atmosphere induced pollution.

### *2.2.3. X-Ray diffraction*

X-Ray diffraction (XRD) patterns were acquired using a Bruker AXS D8 Advance Diffractometer with a Cu tube (40 mA, 40 kV) and a 0.6 mm diffusion slit. Diffracted X-Rays were collected with a Lynxeye detector, between 20 and 60 deg 2θ (since this range includes all main Ti diffraction peaks), with steps of at 1 s/step. XRD peaks identification was performed using Eva software (Brucker). Lattice parameters and phase quantification were obtained after Rietveld Refinement (Topas 4.0 Software, Bruker, Germany). Distortions were quantified through the strain-G parameter, which measures the widening of the Gaussian part of diffraction peaks and is related to the distortion by: strain-G = 4 ε (Topas 4.0 technical reference, Brucker).

### *2.2.4. Mechanical tests*

Three-points bending tests were conducted on 10 constitutive filaments of each condition, using an Electroforce 3200 test machine (Bose, Eden Prairie, MN) equipped with a 22 N load cell. Tests

consisted in a preload of 0.2 N, followed by a loading rate of 0.05N·s$^{-1}$. The length between the two support points was fixed at 10 mm. Since the cross-sectional area may vary slightly from one sample to another, the actual cross-sectional area of each sample was measured using a RH-2000 microscope (Hirox Europe, Limonest, France). The stress-strain curves were then obtained from the load-displacement curves using:

$$\sigma = \frac{FL}{4I_{gz}} y \qquad (1)$$

and:

$$\varepsilon = \frac{6D}{L^2} y \qquad (2)$$

where: $F$ is the force applied at the fracture point, $L$ is the length of the support span, $I_{gz}$ the quadratic moment of the section at the point of rupture, $y$ is the distance to the neutral axis and $D$ the deflection.

Vickers hardness tests were performed under 500 gf loads, on a previously polished scaffold cross-section, perpendicular to the building direction. The average values were calculated on more than 10 measurements.

Compressive tests were conducted on approximately cubic scaffolds, using an Instron 8502, equipped with a 100 kN load cell, and consisted in a preload of 100 N, followed by a strain rate of 0.05 min$^{-1}$. Tested samples were made of 18 layers of 10 struts (~10.2-10.3 mm side after sintering), to follow Ashby's advice on preventing size effects by having at least 7 times the foam cell size in each direction [30]. Upper and lower surfaces were polished prior to test, in order to get flat and parallel surfaces. Stress-strain curves were then obtained after correcting the data from the machine stiffness (measured on a sample with known stiffness: aluminium alloy 2017A at T4 with a Young's modulus of 74 GPa). One compressive test per sample type was interrupted several times at different loads, to enable the evaluation of damage in 3D by X-Ray tomograhy.

### *2.2.5. X-Ray tomography*

3D damage during compressive test was analysed by X-Ray computed tomography. Scaffolds were scanned at different strain levels, using a V|tome|x device (GE Sensing & Inspection Technologies Phoenix X-Ray GmbH, Boston, MA, USA) (0.3 mm Cu filter, 140 kV X-Ray tube voltage, 1200 projections over 360° rotation, 3 images per projection, 667 ms exposure time, 8 µm voxel size).
In addition, sintered mono-filaments printed with a nozzle of 840 µm were imaged at high resolution using an EasyTom Nano tomograph (RX Solutions, Chavanod, France) (100 kV X-Ray tube voltage, 1248 projections over a 360° rotation, 3 images per projection, 4 s exposure time, 0.45 µm voxel size) to analyse porosity.
All the 3D volumes were reconstructed from the collected radiographs using a filtered back projection Feldkamp-algorithm. Images analysis was then performed using the free and open-source Fiji software [31].

### *2.2.6. Density*

X-Ray tomography scans were used to obtain the volume occupied by each scanned scaffold using the same alignment method as described in a previous work [6]. The scanned scaffolds were then weighed to be able to get their density. The Ti64 density used to obtain scaffolds relative density was 4.43 g·cm$^{-3}$.

### *2.2.7. Porosity*

Internal porosity (within filaments) was quantified by analysing X-Ray tomography scans. After binarisation, a mask is obtained from each scan of filaments by filling all pores. The initial binarised scan is then subtracted from this mask to only keep pores. Then pores are labelled (one label per pore) to get dimensional information on each of them, authorising to plot a pore size distribution. Labels containing eight voxels or less are considered as noise and are not kept for characterisations. The volume fraction of porosity is obtained by comparing the number of voxels included in the mask with

the number of voxels included in the labelled pores. In addition, Fiji software is able to detect edges. Thus by adding the edges of the mask to the stack of pores, it is possible to determine which ones are open (in contact with the edge of the mask), and which ones are closed.

### *2.2.8. Scanning electron microscopy*

Images of sample surfaces, cross-sections and fracture surfaces were acquired within a Tescan Vega3 scanning electron microscope (SEM) equipped with a tungsten filament (Tescan Orsay Holding, a.s., Brno, Czech Republic). Both secondary electrons (SE) and backscattered electrons (BSE) modes have been used. In addition, Energy Dispersive X-Ray spectroscopy (EDX) was performed at 20 keV to control chemical composition of the formed phases.
Electron Backscatter Diffraction (EBSD) map acquisitions were performed in a SEM Zeiss Supra 55 VP with field-emission gun, equipped with an Oxford Instrument EBSD Symetry camera. EBSD maps were performed using an acceleration voltage of 20 kV, with an aperture size of 120 µm, in the high current mode. The step size was fixed at 200 nm, in order to well detect the β-Ti phase.
EBSD maps were post-treated using Channel 5 software and a 5° grain boundary criterion to calculate the mean grain size.

## 3. Calculation and modeling: C/O diffusion and precipitation

Precipitation of titanium carbides has been modeled using in-house software PreciSo [32,33], a multi-class, Kampmann-Wagner Numerical (KWN) precipitation model [34] based on Classical Nucleation and Growth Theories (CNGTs). For more details on the implementation on CNGTs, please refer to ref. [35].
Entry parameters of the precipitation model are the temperature profile and the thermodynamic properties of precipitates (chemical composition of precipitates *i.e.* $Ti_6C_4$ from TTTI3 [36] database, surface energy $\gamma$, solubility products $K^{\alpha}_{Ti_6C_4}$ and $K^{\beta}_{Ti_6C_4}$ in $\alpha$ and $\beta$ phases). The output parameter is the precipitate size distribution ($N_i(r_i)$, the number density of precipitates of size $r_i$) and the remaining solute content.

### *Nucleation*

The nucleation rate *dN/dt* gives the flux of precipitates reaching the critical size $R^*$, above which they are stable and grow:

$$\frac{dN}{dt} = N_0^j Z^j \beta^{*j} exp\left[-\frac{\Delta G^{*j}}{k_B T}\right] I(t) \qquad (3)$$

where $j$ stands for either $\alpha$ or $\beta$ Ti matrix, $N_0^j$ is the nucleation site number density, $Z^j$ is the Zeldovich factor, $\beta^{*j}$ is the condensation rate, $\Delta G^{*j}$ is the energy barrier for nucleation, $k_B$ is the Boltzmann constant, $T$ is the temperature and $I(t)$ is an incubation coefficient varying from 0 to 1 with $I(0)=0$ and $I(\infty)=1$.
The energy barrier $\Delta G^{*j}$ depends on surface energy and driving force for precipitation that is calculated from TTTI3 [36] thermodynamic database *via* solubility products.

### *Growth*

The growth equation of spherical precipitates is given solving Fick's diffusion equation assuming a stationary solute concentration profile:

$$\frac{dr}{dt} = \frac{D_C^j}{r}\frac{X_C^0 - X_C^{eq}(r)}{\alpha X_C^P - X_C^{eq}(r)} \qquad (4)$$

where $\alpha$ is the ratio of atomic volumes of matrix and precipitates: $\alpha = {v_{at}^{\alpha/\beta}}/{v_{at}^{P}}$ and $X_C^0$ is the C content and $X_C^{eq}(r)$ are the equilibrium solute concentrations at the interface. They are given by the Gibbs-Thomson equation (see [37,38]):

$$X_{Ti}^{eq\,6} X_{C}^{eq\,4} = K_{Ti_6C_4}^{j} exp\left(\frac{20\gamma v_{at}^{P}}{r}\right) \quad (5)$$

Solubility products were fitted from TTTI3 [36] thermodynamic database for various O contents:

$$K_{Ti_6C_4}^{j} = \frac{C^j}{T^2} - \frac{A^j}{T} + B^j \quad (6)$$

where $A^j$, $B^j$ and $C^j$ are constants depending of the O content of the matrix $[O]_{wt\%}$ *via* $A^j = a_A^j [O]_{wt\%} + b_A^j$, $B^j = a_B^j [O]_{wt\%} + b_B^j$ and $C^j = a_C^j [O]_{wt\%} + b_C^j$.

In this approach, the surface energy $\gamma$ is the only adjustable parameter. All model parameters are summarised in Table 3.

*Table 2. Measured interstitials contents for different conditions associated with their nomenclature.*

| Sample | Debinding conditions | Sintering conditions | [C] (wt.%) | [O] (wt.%) | [N] (wt.%) | [H] (wt.%) |
|---|---|---|---|---|---|---|
| **Powder** | - | - | 0.011 | 0.102 | 0.009 | 0.003 |
| **D500** | 500°C 1°C·min$^{-1}$ vacuum | - | 0.187 | 0.335 | 0.009 | 0.012 |
| **D500-S1200** | 500°C 1°C·min$^{-1}$ vacuum | 1200°C 10°C·min$^{-1}$ vacuum | 0.184 | 0.488 | 0.011 | 0.002 |
| **D350** | 350°C 1°C·min$^{-1}$ vacuum | - | 0.130 | 0.125 | 0.008 | 0.017 |
| **D350-S1200** | 350°C 1°C·min$^{-1}$ vacuum | 1200°C 10°C·min$^{-1}$ vacuum | 0.094 | 0.219 | 0.010 | 0.002 |
| **D350Ar** | 350°C 1°C·min$^{-1}$ Ar flow | - | 0.221 | 0.222 | - | - |
| **D350Ar-S1200** | 350°C 1°C·min$^{-1}$ Ar flow | 1200°C 10°C·min$^{-1}$ vacuum | 0.157 | 0.244 | - | - |
| **5-D350** | 350°C 5°C·min$^{-1}$ vacuum | - | 0.092 | 0.166 | - | - |
| **5-D350-S1200** | 350°C 5°C·min$^{-1}$ vacuum | 1200°C 10°C·min$^{-1}$ vacuum | 0.088 | 0.273 | - | - |

*Table 3. Parameters of the diffusion/precipitation model.*

| Parameter | Value | Unit | Ref. |
|---|---|---|---|
| $D_{0O}^{C/\alpha}$ | $5.06 \cdot 10^{-4}$ | m²/s | [13] |
| $Q_C^{\alpha}$ | 182 | kJ/mol | [13] |
| $D_O^{C/\beta}$ | $1.08 \cdot 10^{-2}$ | m²/s | [13] |
| $Q_C^{\beta}$ | 202 | kJ/mol | [13] |
| $D_O^{O/\alpha}$ | $1.23 \cdot 10^{-7}$ | m²/s | [49] |
| $Q_O^{\alpha}$ | 150 | kJ/mol | [49] |
| $D_O^{O/\beta}$ | $1.6 \cdot 10^{-4}$ | m²/s | [13] |
| $Q_O^{\beta}$ | 201 | kJ/mol | [13] |
| $\gamma^{\alpha}$ | 0.2 | J/m² | Fitted |
| $\gamma^{\beta}$ | 0.2 | J/m² | Fitted |
| $a_A^{\alpha}$ | -1978 | K/(wt%) | [36] |
| $b_A^{\alpha}$ | 7665 | K | [36] |
| $a_A^{\beta}$ | 1241 | K/(wt%) | [36] |
| $b_A^{\beta}$ | 7754 | K | [36] |
| $a_B^{\alpha}$ | -1.827 | 1/(wt%) | [36] |
| $b_B^{\alpha}$ | -0.210 | - | [36] |
| $a_B^{\beta}$ | 0.295 | 1/(wt%) | [36] |
| $b_B^{\beta}$ | -1.593 | - | [36] |
| $a_C^{\alpha}$ | $-9.133 \cdot 10^{5}$ | K²/(wt%) | [36] |
| $b_C^{\alpha}$ | $8.593 \cdot 10^{5}$ | K² | [36] |
| $a_C^{\beta}$ | $5.362 \cdot 10^{4}$ | K²/(wt%) | [36] |
| $b_C^{\beta}$ | $-9.479 \cdot 10^{5}$ | K² | [36] |
| $v_{at}^{P}$ | $1.029 \cdot 10^{-29}$ | m³ | |
| $v_{at}^{\alpha}$ | $1.743 \cdot 10^{-29}$ | m³ | |
| $v_{at}^{\beta}$ | $1.679 \cdot 10^{-29}$ | m³ | |
| $X_C^0\|_{node1}$ | 2 | wt% | Fitted |
| $X_O^0\|_{node1}$ | 15 | wt% | Fitted |

### *Implementations*

At each time step, if the number of nuclei is larger than a critical value, a new class of precipitate is created with number density $N_i = dN/dt \times \Delta t$. Then all existing classes grow according to the solution of the non-linear system formed by Eqs. (4) and (5), leading to $r_i(t+\Delta t) = r_i(t) + dr/dt \times \Delta t$. Finally, the mass balance is performed to update the solute content of all solute species.

### *Coupled diffusion/precipitation model*

In its most recent version called NodePreciso [24], [25] (based on the contributions of Gouné et al [23] and Van Landeghem et al [39]), the software has been improved to be able to model precipitation within a chemical gradient of diffusing solute atoms. At the beginning of the simulation a mesh is defined.

In this simulation, the mesh consisted in 3 nodes. [node 0] and [node 2] represent the bulk and the surface of a powder particle (of size 25 µm), respectively, whereas [node 1] represents a C/O rich layer of the binder (of thickness 1 µm) (see Figure 1(a)). All nodes have a given initial solute content $X^0$ and are connected to their neighbouring nodes via a given surface. Each time step of the simulation starts with a diffusion step during which a node can exchange atoms with its neighbours. Fluxes of

atoms are given by Fick's law. Note that precipitation is allowed to occur only in the Ti particle bulk [node 0].

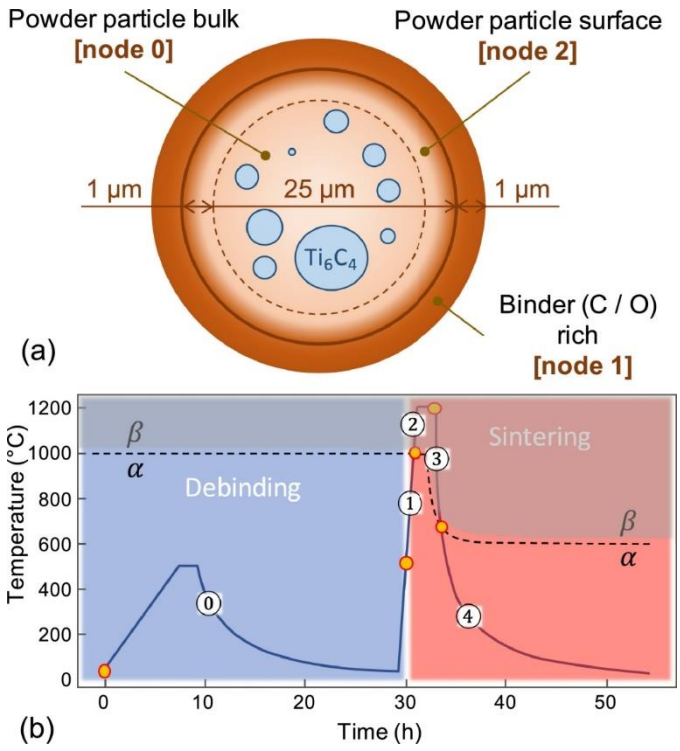



*Figure 1. (a) Simulated Ti64 powder grain: the precipitation model is applied on 3 nodes exchanging solute atoms: [node 0] represents the bulk of a grain of Ti64 powder, [node 1] represent a C/O rich layer issued from the binder and [node 2] is the surface layer of the Ti grain. (b) Simulated thermal profile composed of a debinding stage [stage ⓪], heating to the α/β transformation temperature [stage ①], heating and holding at the sintering temperature [stage ②], cooling from sintering temperature to the β/α transformation temperature [stage ③] and cooling in α phase to room temperature [stage④].*

*Thermal treatments*

Thermal treatments are modelled by a succession of 5 stages: a debinding stage [stage ⓪], heating to the $\alpha/\beta$ transformation temperature $T_{\alpha\beta}$ [stage ①], heating and holding at the sintering temperature $T_S$ [stage ②], cooling from sintering temperature to the $\beta/\alpha$ transformation temperature $T_{\beta\alpha}$ [stage ③] and cooling in $\alpha$ phase to room temperature [stage ④] (see Figure 1 (b)). Note that $T_{\beta\alpha}<T_{\alpha\beta}$, because the cooling rate is high compared to the transformation kinetics, leading to a shift in the transformation temperature [40].

The Python Notebook used to perform all diffusion and precipitation simulations is provided as supplementary materials at the end of this document.

## 4. Results

This study aimed at understanding the influence of processing parameters on the development of the microstructure of 3D printed Ti64 alloys, in particular from the point of view of oxygen and carbon enrichment, and to correlate it with mechanical properties.

### 4.1. Set up of debinding thermal cycles

Thermogravimetric analysis (TGA) was realised on dried sample to be as close as possible to the debinding conditions. Thus, the only observed mass loss on Figure 2 is due to the pluronic F-127 degradation. In addition, the ink contains 5 wt% of pluronic F-127, meaning that the debinding step is fully completed at 350 °C. Thus, even though previous parts printed with pluronic F-127 were usually debinded at 500 °C or above [10,27,29], a temperature of 350 °C should be enough. Consequently, two debinding cycles were studied here: 30 min at 350 °C (minimal temperature for a complete debinding), and 2h at 500 °C (usual debinding cycle).

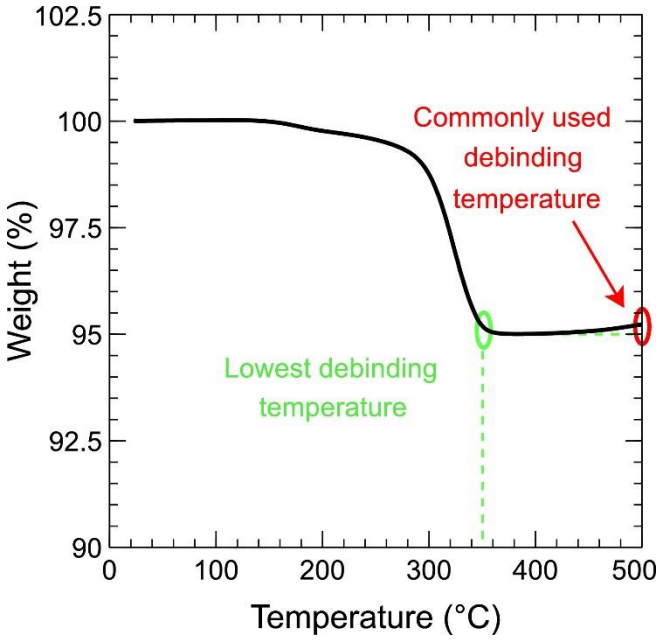



*Figure 2. Thermogravimetric analysis (TGA) with the two debinded temperatures studied.*

TGA was performed under nitrogen flow, but titanium might easily interact with nitrogen to form titanium nitride, which might be the reason of the small mass gain observed above 450°C on Figure 2. Thus debinding treatments were performed under dynamic vacuum or argon flow instead of nitrogen flow. The atmospheres chosen (vacuum or argon) for debinding are inert and will therefore not influence the degradation of the binder (temperature and kinetics comparable to the measurement carried out under nitrogen).

### 4.2. Resulting interstitials contents

Debinding 2 h at 500 °C or 30 min at 350 °C give rise to carbon and oxygen uptake (Table 2). Also, the decrease in carbon content after sintering for all samples debinded at 350 °C (compared to their as debinded state) suggests that a small amount of binder remains after debinding at 350 °C. This seems confirmed by hydrogen content evolution for both debinding temperatures: after debinding (for both conditions) a small amount of hydrogen remains. It completely disappears after sintering. The only source of hydrogen being the binder, it can be inferred that whatever the debinding temperature used, it is the sintering step which makes it possible to get rid of the binder completely.

Another important point is the carbon content influence on oxidation. While carbon uptake only takes place during debinding (as the only carbon source is the binder), oxidation occurs at each step. And oxidation during sintering might be limited by the presence of carbon, as shown by D350Ar-S1200 samples. Indeed, at 1200 °C under vacuum, carbon is able to reduce titanium oxide [41], and this positive effect of a high carbon pick-up on further oxidation control, has already been observed in powder injection moulding of titanium [14].

### 4.3. Microstructural analysis

Figure 3 shows XRD patterns for debinded and sintered samples in comparison with the initial powder. Different debinding parameters were compared but all the debinded samples were submitted to the same sintering treatment of 2 h at 1200 °C under secondary vacuum.

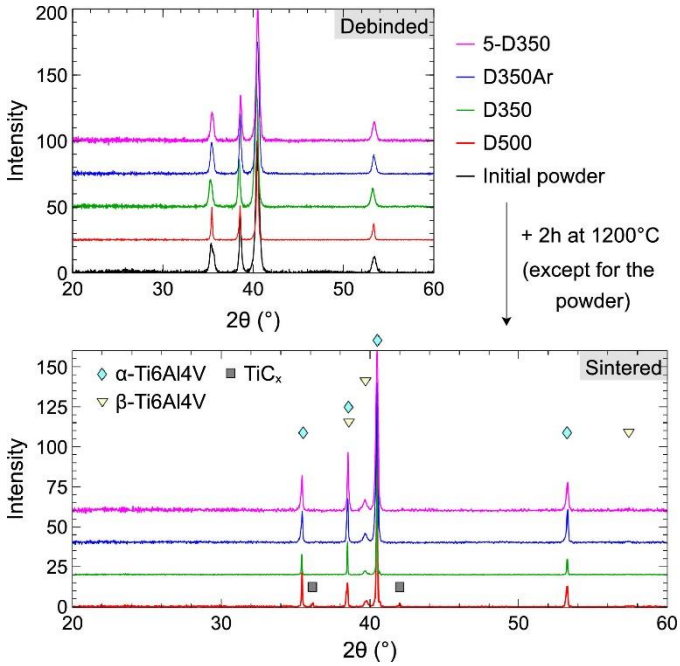



*Figure 3. Evolution of XRD patterns after debinding and sintering for different debinding conditions.*

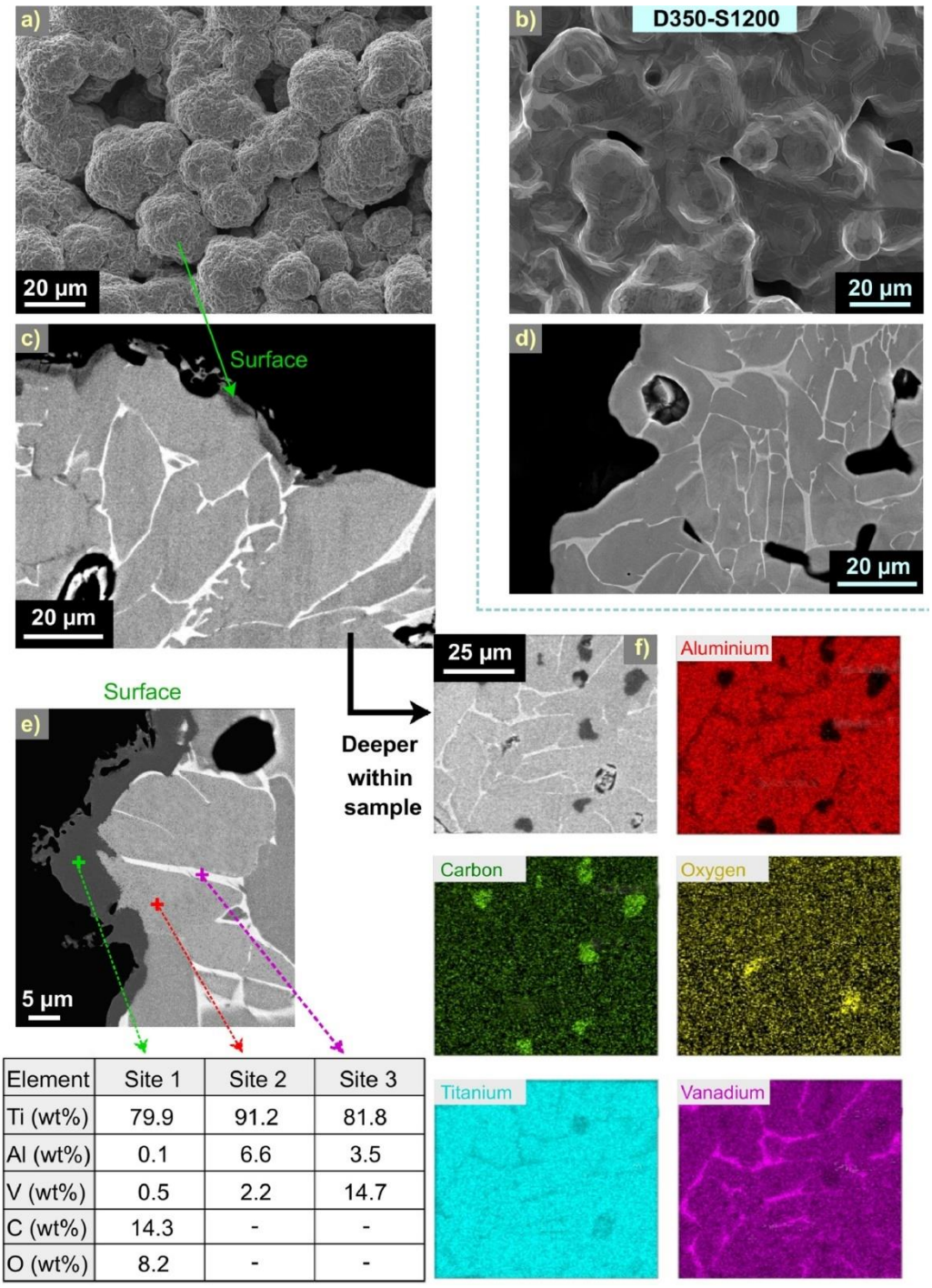
XRD patterns of initial powders and debinded samples show the presence of the α-titanium phase only (Powder Diffraction File (PDF) 00-044 1294), indicating that there was no crystallographic transformation during the debinding step, regardless of the debinding conditions.

Debinding at 500 °C leads to higher and thinner peaks than debinding at 350°C. This can be explained by the recovery phenomenon induced by the debinding treatment. Indeed, gas-atomised powders have fine martensitic structure with a potential high dislocations density, which debinding treatment at least partially decreases. A treatment of 2 h at 500°C eliminates more dislocations than a 30 min treatment at 350°C. This is confirmed by the decrease of the "strain-G" parameter (obtained from Rietveld refinement) with increasing thermal treatment temperature.

After sintering, α-phase remains predominantly present within all samples, together with around 10 wt% of β-titanium phase (estimated by XRD, using lattice parameters compatible with those given by Malinov et al. [42]). After sintering at 1200 °C, samples debinded at 350 °C contain no other phase. However, D500-S1200 samples contain also a small amount of titanium carbides. These peaks can be fitted both by considering the cubic TiC lattice (PDF 01-1222) as well as by considering the cubic $TiC_{0.62}$ lattice (cif file number 1532224 from the crystallography open database) and the hexagonal $Ti_6C_{3.75}$ lattice (cif file number 1540227). These last two lattices are in agreement with the chemical composition given as thermally stable by Thermocalc TTTI3 database ($Ti_6C_4$).

Thus precipitates are only detectable by XRD in D500-S1200 samples after sintering. This means that in others conditions carbides are either non-existent or too small or too few in weight to be detectable. Indeed, 3 wt% of TiC smaller than 30 nm gives a signal of the order of background noise magnitude.

To get a better overview of these precipitates, surfaces and cross-sections of both D350-S1200 and D500-S1200 have been imaged by SEM as shown on Figure 4. Surface structure of the two samples is completely different. D500-S1200 surface has a granular appearance with "granule" sizes smaller than the size of Ti64 particles (Figure 4(a)). This might be due to the formation of a contaminant layer on the surface as suggested by the cross-section (Figure 4(c)). In comparison, D350-S1200 surface is much smoother, titanium particles are directly visible. However, it should be noted that they present faceted surfaces, probably due to anisotropic surface diffusion as explained by Joo et al. study [43]. In addition, cross-section shown on Figure 4(d) confirms that there is no additional phase formation on D350-S1200 surface. Figure 4 (e) focuses on the surface layer of D500-S1200 sample. This layer can be almost 10 μm thick in some areas. However, its thickness is not homogeneous and the layer even presents discontinuities as shown on Figure 4 (c).




| Element | Site 1 | Site 2 | Site 3 |
|---|---|---|---|
| Ti (wt%) | 79.9 | 91.2 | 81.8 |
| Al (wt%) | 0.1 | 6.6 | 3.5 |
| V (wt%) | 0.5 | 2.2 | 14.7 |
| C (wt%) | 14.3 | - | - |
| O (wt%) | 8.2 | - | - |

*Figure 4. Surface aspect of (a) D500-S1200 and (b) D350-S1200; Cross-section showing surface of (c) D500-S1200 and (d) D350-S1200 ; EDX analysis showing carbide formation on surface (e), as well as in the bulk (f) (See Table 2).*

EDX analysis shown on Figure 4(e) suggests that this is a carbon rich layer which might also contain oxygen. In addition to this carbon rich layer, titanium carbides can also be found in D500-S1200 bulk (Figure 4(f)), whereas higher oxygen concentration were only detected in sample pores. Also, it should be noted that no carbides were observed in D350-S1200 bulk, and that carbides in D500-S1200 bulk were non-homogeneously distributed.

To complete the microstructural study, Figure 5 shows inverse pole figure maps obtained for both D350-S1200 and D500-S1200 samples, with their associated phase maps. First it can be observed that D500-S1200 presents, in average, twice smaller $\alpha$ grains than D350-S1200. In addition, these smaller $\alpha$ grains are fully equiaxed, whereas coarser grains of D350-S1200 are still lamellar.

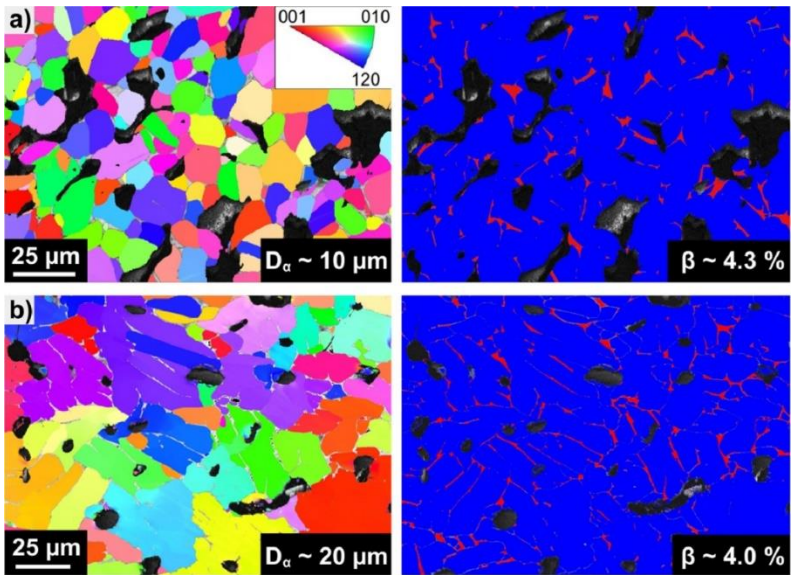



*Figure 5. As-sintered microstructure : EBSD inverse pole figure map and its associated phase map of: (a) D500-S1200 and (b) D350-S1200 samples.*

Also, the difference in $\alpha$ grains morphology goes with a difference in morphology of the inter-granular beta phase but does not seem to change the $\alpha/\beta$ ratio, with an estimation through EBSD of 4 to 4.5 % remaining $\beta$ phase. The difference between this value and the one estimated through XRD measurements can be explained by several factors. First, $\beta$ being the finest phase, it may be less well indexed and therefore slightly underestimated by EBSD. In addition, EBSD is a surface analysis carried out on a relatively small area. Finally, XRD quantification can lead to an overestimation of the $\beta$ phase as the presence of aluminium in the $\alpha$ phase and vanadium in the $\beta$ phase were not taken into account and preferential orientation effects were neglected.
Finally, it should be noted that the chosen sintering treatment leads to a residual porosity within filaments of around 12 - 13 %, regardless of the debinding conditions used. As illustrated by the Fig. 7 of our precedent work [6], this porosity is composed of 97 % of open and interconnected porosity and only 3 % of closed pores. Combining the macropores of the design with these residual micropores results in scaffolds with an average density of 57.3 %.

## 4.4. Mechanical properties

### *4.4.1. Bending properties of constitutive filaments*

Three-points bending tests on constitutive filaments enable to determine the intrinsic mechanical properties of the printed material (without structure effect). These tests were performed on the three conditions D500-S1200, D350-S1200 and D350Ar-S1200, which represent the two extreme amounts of interstitials obtained and an intermediate case among the sintered materials.
Stress-strain curves presented on Figure 6 show that D500-S1200 samples are fully brittle whereas the two other conditions lead to a bit of ductility. The ultimate stress decreases with increasing interstitials amount. As a result, the ultimate strength of the least brittle sample (D350-S1200) is twice higher than that of the most brittle one (D500-S1200).
While the use of 0.2 $L.min^{-1}$ argon flow does not lead to mechanical strength as high as the dynamic primary vacuum due to higher oxygen and carbon contamination, it seems to reduce scattering of the results. Thus it would be interesting to check whether increasing the flow would further reduce carbon and oxygen pollution or not.
Young's moduli measured with the stress-strain curves are 60 ± 9 GPa, 70 ± 9 GPa and 68 ± 7 GPa for D350-S1200, D350Ar-S1200 and D500-S1200 respectively. Considering the high standard deviation, no obvious influence of debinding conditions on the Young's modulus can be determined here.
Fracture surfaces shown on Figure 7 confirm both: the brittle behaviour of D500-S1200 samples which are fully cleaved, and the mixed brittle-ductile behaviour of samples debinded at 350 °C which present dimples in addition to cleavage. In D500-S1200, titanium carbides can be found in cleavage

sites (Figure 7(d)). Carbides could not be detected at cleavage sites of samples debinded at 350 °C (Figure 7(c)), neither by XRD nor by SEM.

In addition to three-points bending tests, Vickers hardness was measured for D350-S1200 and D500-S1200 samples. The load used was sufficiently low (500 gf) to characterise only filaments and not whole structure. D500-S1200 samples present a higher hardness than D350-S1200 samples, with an average value of 298±16 $HV_{0.5}$ compared to 243±8 $HV_{0.5}$ for D350-S1200 samples.

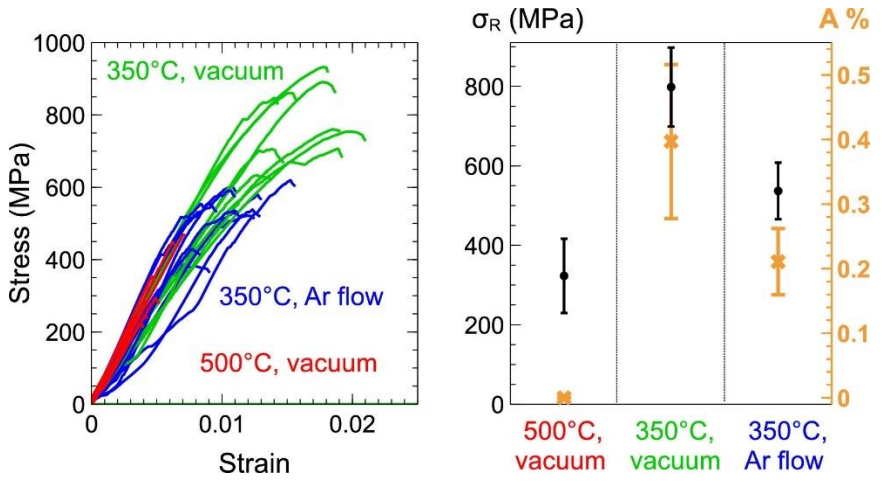
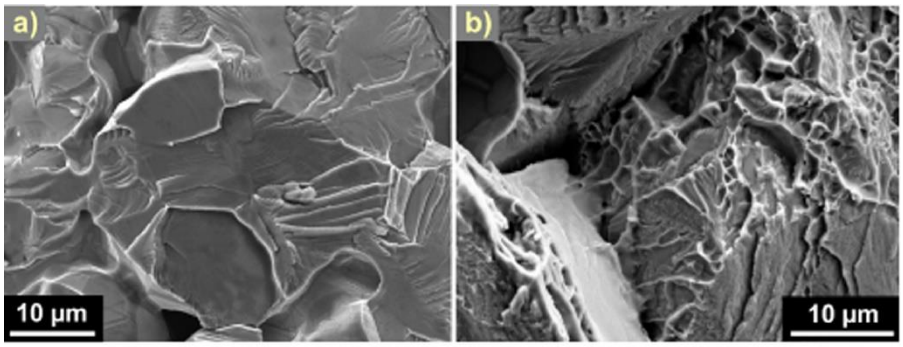
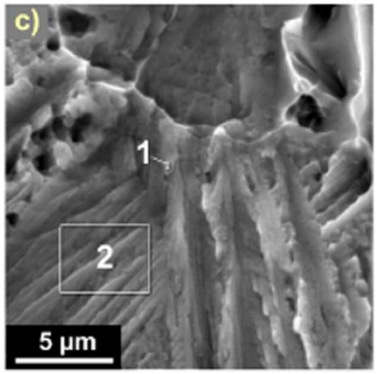
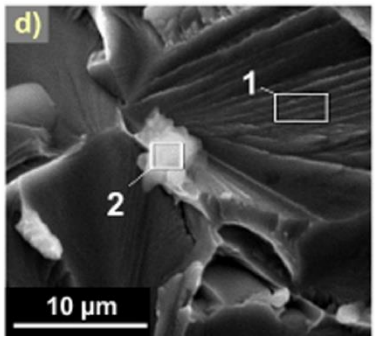



*Figure 6. Three-points bending tests: Stress-strain curves obtained for D500-S1200, D350Ar-S1200 and D350-S1200 with their associated ultimate stress and elongation at break.*




| Element | Site 1 | Site 2 | Whole |
|---|---|---|---|
| Ti (wt %) | 93.3 | 94.9 | 92.9 |
| Al (wt %) | 5.3 | 3.7 | 4.0 |
| V (wt %) | 1.4 | 1.4 | 3.1 |




| Element | Site 1 | Site 2 | Whole |
|---|---|---|---|
| Ti (wt %) | 94.9 | 58.9 | 89.2 |
| Al (wt %) | 2.9 | 2.3 | 4.4 |
| V (wt %) | 2.2 | 2.0 | 3.8 |
| C (wt %) | - | 36.8 | 2.6 |

*Figure 7. Fracture surfaces: (a) Surface of D500-S1200 showing brittle behaviour (cleavage); (b) Surface of D350-S1200 showing mixed ductile-brittle behaviour (cleavage and dimples); (c) Focus on a cleavage site of D350Ar-S1200 showing standard composition for Ti64 (EDX analysis); (d) Focus on a cleavage site of D500-S1200 showing the presence of a carbide.*

### *4.4.2. Compression tests on scaffold structures*

Compression tests were performed on scaffolds from the two extreme debinding conditions, D500-S1200 and D350-S1200. Stress-strain curves are shown on Figure 8.

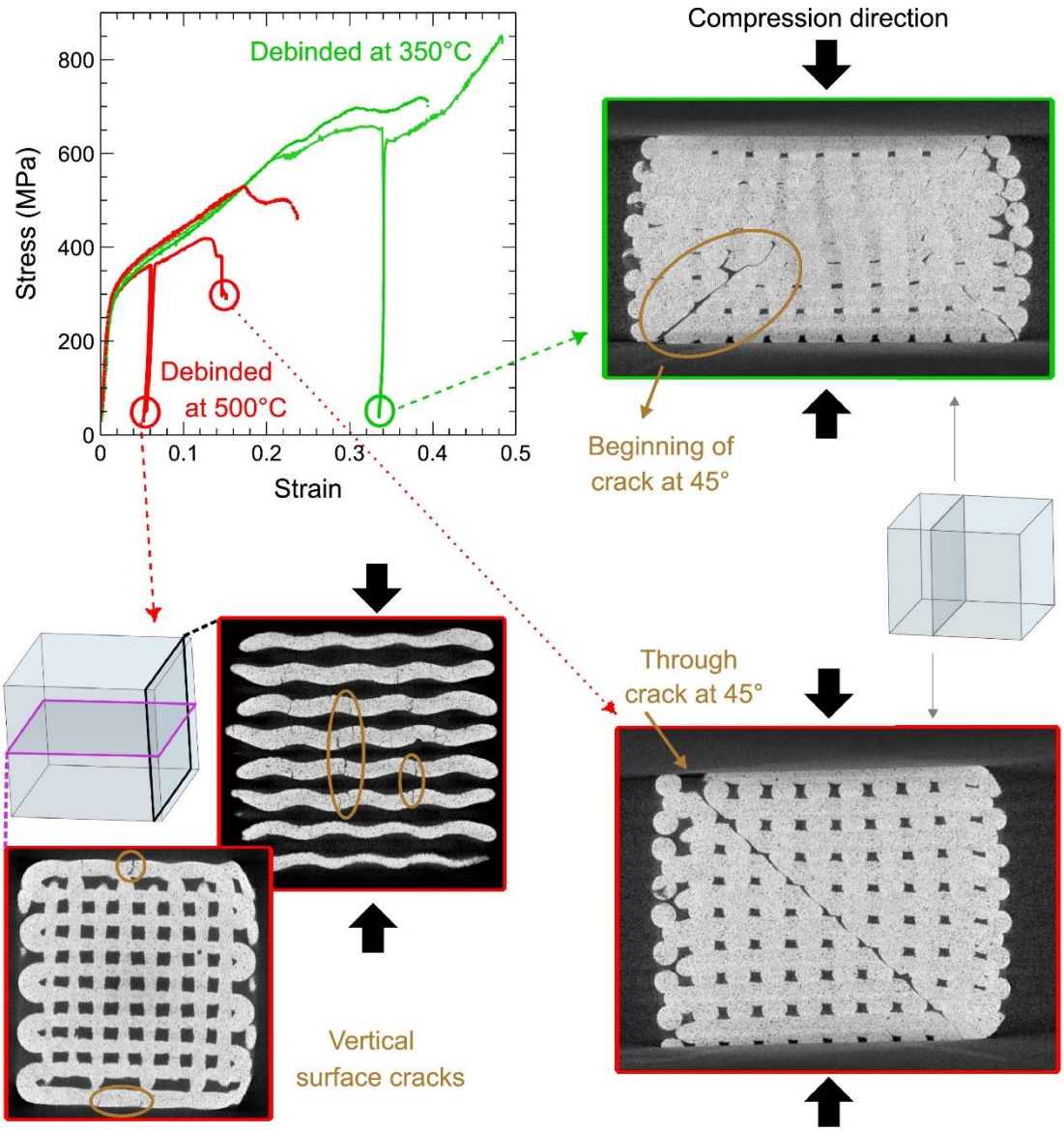



*Figure 8. Compressive tests : Stress-strain curves obtained for both D500-S1200 and D350-S1200; X-Ray tomographic sections of both cases showing internal damage for different strain levels.*

First it can be observed that the global behaviour of the two kinds of samples is similar. The elastic domain is followed by densification during which vertical cracks are formed on sample surfaces as shown for D500-S1200 on Figure 8. This phenomenon has been previously observed on D350-S1200 [6]. Then an internal crack is initiated and spreads at 45° throughout the sample. This crack is initiated earlier and propagates faster for the D500-S1200 scaffolds than for D350-S1200 as highlighted on Figure 8. For D350-S1200 samples, tests were stopped at 90 kN to prevent damage of the 100 kN load cell, thus ultimate strengths were not measured.

Both D350-S1200 and D500-S1200 printed structures present similar apparent Young's modulus (around 30 GPa) and yield stress (around 250 MPa). Although the yield stress of D500-S1200 seems to be a bit lower and more scattered than the one of D350-S1200 (246 *vs* 265MPa), more statistic is needed to confirm this difference.

## 5. Discussion

### 5.1. C/O uptake

The two pairs of debinding parameters (30 min at 350 °C and 2 h at 500 °C) lead to very different carbon and oxygen enrichments, as shown in Table 2. These values are the result of C/O diffusion from the binder to the Ti64 particles during the debinding treatment. In order to verify the consistency of these C and O uptakes, diffusion simulations were performed on a Ti64 powder particle (bulk: [node0] and surface: [node2]) coated with C and O rich binder layer [node1] (see Figure 1(a)). All parameters used in this simulation are recalled in Table 3.

As a first step, no precipitation was permitted within the Ti64 particle and only diffusion of C/O was allowed. Figure 9 compares the predicted values of total C and O content in the powder particle ([node0] and [node2]) with the amount measured by spectroscopy for D350, D350-S1200, D500 and D500-S1200 samples.

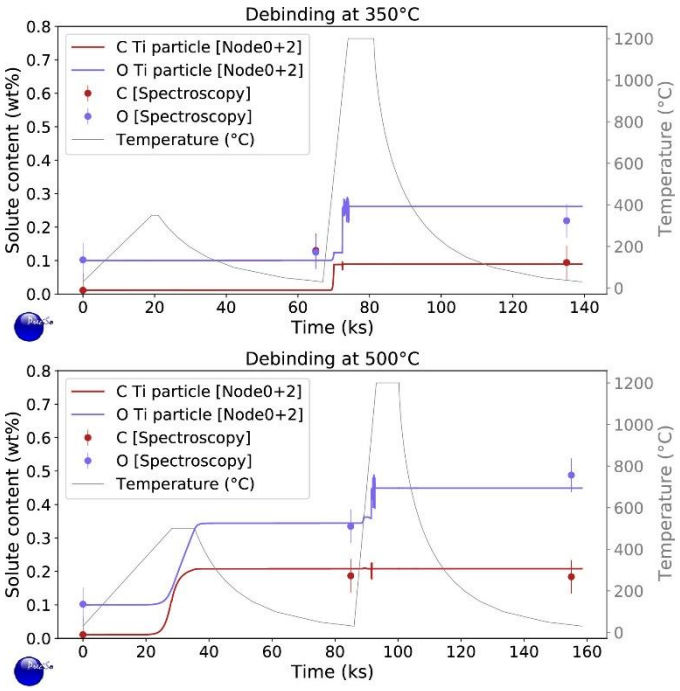



*Figure 9. Total amount of C and O in a Ti64 particle [node0]+[node2]) during the debinding and sintering processes. The model is based on a C/O rich layer present during debinding [node1] (see Figure 1(a)). The simulated diffusion is compared to the C and O values obtained by spectroscopy for samples D350, D350-S1200, D500 and D500-S1200 (heat treatments under vacuum).*

Starting from the initial C/O contents measured in the powder, the simulation shows a fast enrichment in C and O during the debinding treatment at 500 °C and no increase in C and O content during the debinding at 350 °C. The presence of remaining binder noticed after this debinding treatment at 350 °C explains the higher amount of measured C compared to the simulated one. As the C can diffuse from this remaining binder to the Ti64 particle during the ramp up of the sintering treatment between 350 and 500 °C, simulated and measured C contents agree after sintering. Note that O enrichment of Ti64 particles during the sintering stage at 1200 °C is attributed to the presence of the quartz ($SiO_2$), which could act as an O source at such high temperatures.

This first step of simulation is in agreement with experimental values and can therefore quantitatively explain the C and O uptakes during debinding and sintering treatments.

### 5.2. Carbides precipitation

Two conditions must be met to enable the carbides precipitation: the carbon concentration must be above the carbon solubility limit in Ti64, and the time or temperature must be long or high enough to allow diffusion. According to the XRD patterns, these two conditions are only met for D500-S1200 samples. However carbon solubility limit at 1200 °C in $\beta$-Ti64 containing 0.5 wt% of oxygen is about 0.32 wt% [36]. Thus, global carbon concentration measured in all the sintered samples studied here, including those debinded at 500 °C, is below this solubility limit (see Table 2). In addition, carbon

solubility in the $\alpha$ phase is decreasing below 0.16 wt% at 500 °C. But at this temperature the mobility is probably not sufficient to allow precipitation as no carbides are observed in D500 samples. Thus precipitation may occur during the cooling ramp but in the $\beta$–phase.

Discontinuities of the surface layer present on D500-S1200 added to the surface topology are more reminiscent of a precipitation phenomenon than a simple oxidation. In contrast with bulk carbides that present globular shapes to reduce their interface energy, these surface precipitates keep more elongated shapes, as they only have one side in contact with Ti64 matrix. Also, it should be noted that carbides observed in D500-S1200 bulk were non-homogeneously distributed. This seems in agreement with the hypothesis of a precipitation in $\beta$–phase during the cooling ramp, in the dual phase temperature range.

The presence of carbides is also highlighted by the morphology differences of the α–grains. Indeed, the smaller α–grains of D500-S1200 samples (Fig. 5(b)) can be explained by the presence of titanium carbides, which can limit grain growth due to Zener pinning [44] or act as nucleation sites for the α–grains during the $\beta$–$\alpha$ transformation [45,46]. In addition, their equiaxed morphology is also induced by the presence of titanium carbides as shown in former studies [45,47,48]. Note that these two phenomena cannot be attributed to the presence of oxygen, since an ongoing study with finer powders having a carbon content similar to samples D350-S1200 but a higher oxygen content than samples D500-S1200 reveals a still lamellar microstructure after sintering with a slight magnification of the alpha grains compared to samples D350-S1200.

In order to verify the observations (no precipitation during debinding stages and precipitation of carbides during the sintering treatment for D500-S1200 samples), precipitation simulations were performed on the same system as previously (α Ti64 powder particle (bulk: [node 0] and surface: [node 2]) coated with C and O rich binder layer [node 1] (see Figure 1(a))).

In this second simulation step, precipitation was allowed to occur within the bulk of the Ti64 particle [node 0]. Figure 10 presents the evolution of precipitate volume fraction during the debinding and sintering treatments for both debinding temperatures 350 and 500 °C. Not surprisingly, no precipitation occurs during debinding stages, neither at 350 °C, nor at 500 °C. For both holding temperatures, the solubility limit of C is higher than the maximum amount of C present in the Ti64, so that no precipitation can occur. Precipitation is not observed either during the heating ramp of the sintering treatment, even if there is a driving force for precipitation. Indeed, the nucleation is not occurring due to the low mobility of C: the dynamic evolution of temperature does not give enough time for precipitation to occur.

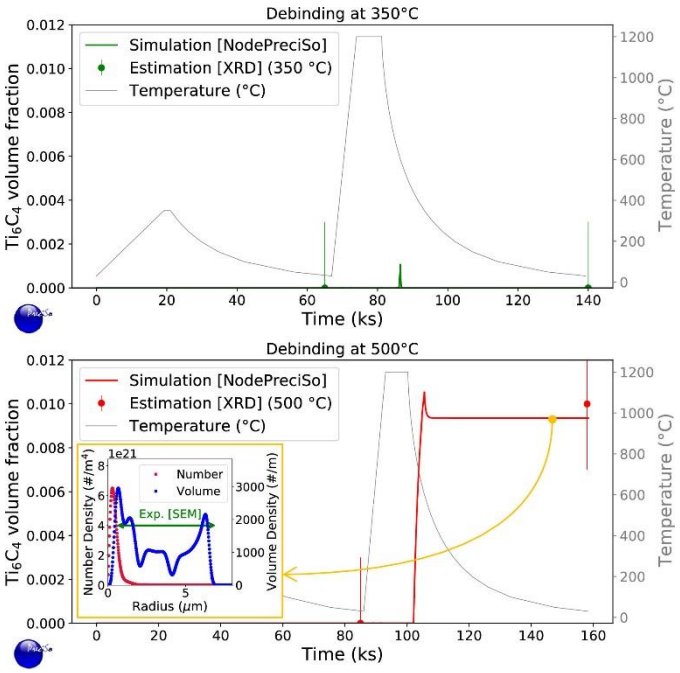



*Figure 10. Evolution of $Ti_6C_4$ precipitates volume fractions during the debinding and sintering processes. Precipitation is predicted during cooling from the sintering stage in the Ti-β phase of the sample debinded at 500 °C (inset: number and volume precipitate size distributions at the end of debinding/sintering treatment are in agreements with SEM observations - see Figure 4).*

At 1200 °C, precipitation is still not observed because the solubility limit of C in the Ti-$\beta$ phase is once again too high (0.32 wt%) compared to the C content (0.2 wt%). However, during the cooling, as the $\beta \rightarrow \alpha$ transition temperature is lowered by the transformation kinetics (as shown in the CCT diagram of Ti-6Al-4V of Dabrowski [40]), a temperature domain ranging from 1000 to 700 °C exists where mobility and driving forces are high enough for precipitation of $Ti_6C_4$ carbides to occur during cooling in the Ti-$\beta$ phase.

Note that only the sample debinded at 500°C exhibits a non-zero volume fraction of precipitates, as observed experimentally. This is because this sample was more enriched in C during the debinding stage. In the inset of Figure 10, it can be observed that the predicted precipitate size distribution is in agreement with the value of 5 µm estimated from SEM characterisation (see Figure 4).
From the particle size distribution, it is possible to calculate Zener [50] and Rios [51] critical grain sizes, giving 250 µm and 500 µm, respectively. These values are much larger than the actual Ti64-$\alpha$ grain size (approx. 10 µm), confirming that TiC precipitates do not act as pinning particles, but rather as nucleation sites for the $\alpha$–phase during cooling, as also supposed in other studies [45,46].
These precipitation simulations are fully consistent with all experimental results and therefore validate the following scenario: (i) C uptake during debinding at 500 °C and (ii) precipitation of carbides in the $\beta$ phase during the cooling after sintering, before and during the $\beta \rightarrow \alpha$ phase transformation.

### 5.3. Mechanical consequences

The mechanical properties of Ti64 samples obtained by DIW depend on their oxygen and carbon enrichments and resulting microstructures. As mentioned in Section 4.2 a carbon enrichment during debinding can help to prevent oxidation during sintering. But as both carbon and oxygen can affect ductility, a compromise between carbon uptake and oxidation should be found. Thus the use of an equivalent oxygen content, calculated according to the contribution of each interstitial element on mechanical properties, might be helpful. This equivalent oxygen content can be defined as [52]:

$$O_{eq} = [O](wt\%) + 2[N](wt\%) + 0.66[C](wt\%) \qquad (7)$$

As the range of measured nitrogen concentration is really narrow, $O_{eq}$ values were calculated using a concentration of 0.01 wt% of nitrogen for all samples for which no measurement was performed. This equivalent oxygen content enables to conclude that D350-S1200 samples represent the best compromise (with a value of 0.301 wt%), which is in agreement with bending tests. Also elongation at break is usually represented as a function of equivalent oxygen content to try to define an acceptable limit. This is for example well used in studies on metal injection molding process [7,14,52].

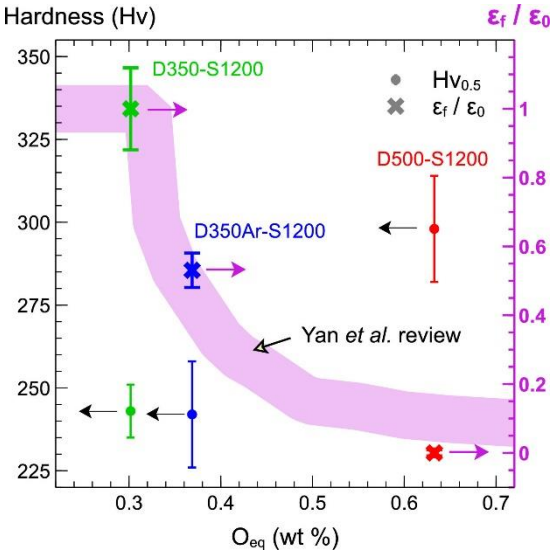



*Figure 11. Hardness and elongation at break evolution as a function of equivalent oxygen content. Elongation at break reported in Yan et al. review is represented for comparison [52].*

In order to get rid of the differences in absolute values obtained due to the difference in porosity of the tested samples, behaviours obtained in this study were compared to behaviours obtained by MIM by plotting $\varepsilon_f / \varepsilon_0$, $\varepsilon_0$ being the elongation at break obtained for the most ductile sample. This enables to highlight the similarity of behaviours with a ductility drop for an equivalent oxygen content around 0.34wt% (see Figure 11).

In addition, Vickers hardness measured on D350-S1200 samples ($HV_{0.5}$ of 243 ±8) is in agreement with values already observed on Ti64 samples with similar sintering density [53-55]. However, D500-S1200 sample presents a higher hardness, 298±16 $HV_{0.5}$, which can be attributed to the higher equivalent oxygen [14].
The presence of carbides in cleavage sites shown in Figure 7(d) highlights their role in initiating the cleavage, in agreement with cracks location observed in Ti64/TiC composites [56]. However, oxygen in solid solution is also known to deteriorate the ductility of titanium and most likely play a role on embrittlement as well [52].

In light of both bending tests on constitutive filaments and compressive tests, scaffold structures allow to recover some ductility. However, the difference in interstitial amount between D500-S1200 and D350-S1200 scaffolds still leads to an important difference in ductility. Thus debinding temperature used after DIW of titanium should always be chosen as low as possible.

The ASTM F1108 standard for Ti64 alloy castings for surgical implants requires the following maximum contents: 0.20 wt% of oxygen, 0.05 wt% of nitrogen and 0.10 wt% of carbon. Thus the D350-S1200 group matches the requirement for carbon and nitrogen but is slightly above the acceptable oxygen content. However, in this work, efforts to minimise interstitials enrichment were only studied during debinding treatments whereas it has been demonstrated that every step (including the selection of the initial powder) plays a part in interstitial enrichment [14]. Thus it might be possible to reduce the oxygen content below this acceptable maximum by playing on initial powder, minimising the binder content, maximising the furnace load during sintering and using yttria or zirconia sample holders instead of quartz. Another interesting point would be to check whether an increase in argon flow would lead (or not) to a better debinding, compared to what has been measured here.

## 6. Conclusion

Interstitial content measurements, thermogravimetric, XRD and SEM analyses and mechanical tests were performed on Ti64 structures obtained using various debinding treatments after direct-ink writing. In parallel, a diffusion/precipitation model was applied to study C and O uptake and titanium carbides precipitation. This combination made it possible to show that:

- 350 °C is the lowest temperature that can be used to eliminate Pluronic F-127 from the 3D printed structures.
- Titanium carbides are only present in samples debinded at 500 °C, which contain the highest interstitial contents. This presence explains the resulting microstructure, with smaller and equiaxed $\alpha$–grains.
- For high enough carbon uptake, titanium carbide precipitation can occur in the $\beta$–phase during cooling from the sintering temperature.
- The loss in ductility observed for an oxygen equivalent around 0.34 wt% is in agreement with previous observations made on Ti64 parts obtained by MIM.
- The scaffold structures enable to recover some ductility compared to the behaviour of constitutive filaments.
- The difference in carbon and oxygen intakes caused by debinding conditions variation is quantitatively explained by diffusion. Therefore, the debinding temperature used after DIW of titanium should always be chosen as low as possible to minimise interstitials diffusion from the binder to the particles.
- In this study, the best compromise in terms of carbon and oxygen uptake is obtained for a debinding treatment of 30 min at 350 °C under dynamic vacuum, using a heating rate of 1 °C.min$^{-1}$. This treatment leads to C and N contents in agreement with the ASTM F1108 standard for Ti64 alloy castings for surgical implants. The oxygen content remains a little bit too high compared to the 0.20 wt% required by the standard but could still be decreased by optimizing the sintering conditions and/or the interstitials contents of the powder used.

## 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.

## References


1. Q. Chen, G.A. Thouas, **Metallic implant biomaterials,** Mater. Sci. Eng., 87 (2015), pp. 1-57, 10.1016/j.mser.2014.10.001
2. C. Leyens, M. Peters, **Titanium and Titanium Alloys: Fundamentals and Applications,** John Wiley & Sons (2003); 10.1002/3527602119
3. J.I. Cesarano, T. Baer, P. Calvert, **Recent Developments in Freeform Fabrication of Dense Ceramics from Slurry Deposition,** Technical Report SAND–97-2857C, CONF-970888–, 554831 (1997), 10.2172/554831
4. J.E. Smay, J. Cesarano, J.A. Lewis, **Colloidal inks for directed assembly of 3-D periodic structures,** Langmuir, 18 (14) (2002), pp. 5429-5437, 10.1021/la0257135
5. J.A. Lewis, **Direct ink writing of 3D functional materials,** Adv. Funct. Mater., 16 (17) (2006), pp. 2193-2204, 10.1002/adfm.200600434
6. M. Coffigniez, L. Gremillard, S. Balvay, J. Lachambre, J. Adrien, X. Boulnat, **Direct-ink writing of strong and biocompatible titanium scaffolds with bimodal interconnected porosity,** Addit. Manuf., 39 (2021), p. 101859, 10.1016/j.addma.2021.101859
7. T. Ebel, **Metal injection molding (MIM) of titanium and titanium alloys,** Handbook of Metal Injection Molding, Elsevier (2012), pp. 415-445, 10.1533/9780857096234.4.415

8. J. Franco, P. Hunger, M. Launey, A. Tomsia, E. Saiz, **Direct write assembly of calcium phosphate scaffolds using a water-based hydrogel,** Acta Biomater., 6 (1) (2010), pp. 218-228, 10.1016/j.actbio.2009.06.031
9. S. Eqtesadi, A. Motealleh, P. Miranda, A. Lemos, A. Rebelo, J.M. Ferreira, **A simple recipe for direct writing complex 45S5 Bioglass® 3D scaffolds,** Mater. Lett., 93 (2013), pp. 68-71, 10.1016/j.matlet.2012.11.043
10. C. Petit, S. Meille, E. Maire, L. Gremillard, J. Adrien, G.Y. Lau, A.P. Tomsia, **Fracture behavior of robocast HA/-TCP scaffolds studied by X-ray tomography and finite element modeling,** J. Eur. Ceram. Soc., 37 (4) (2017), pp. 1735-1745, 10.1016/j.jeurceramsoc.2016.11.035
11. S. Banerjee, C. Joens, **Debinding and sintering of metal injection molding (MIM) components,** Handbook of Metal Injection Molding, Elsevier (2012), pp. 133-180, 10.1533/9780857096234.1.133
12. T. Ebel, O. Milagres Ferri, W. Limberg, M. Oehring, F. Pyczak, F.P. Schimansky, **Metal injection moulding of titanium and titanium-aluminides,** Key Eng. Mater., 520 (2012), pp. 153-160, 10.4028/www.scientific.net/KEM.520.153
13. H. Ogden, R. Jaffee, **The Effects of Carbon, Oxygen, and Nitrogen on the Mechanical Properties of Titanium and Titanium Alloys,** Technical Report TML-20, 4370612 (1955), 10.2172/4370612
14. E. Baril, **Titanium and titanium alloy powder injection moulding: matching application requirements,** PIM Int., 4 (4) (2010), pp. 22-32
15. J.P. Li, J.R. de Wijn, C.A. van Blitterswijk, K. de. Groot, **Porous Ti6Al4V scaffolds directly fabricated by 3D fibre deposition technique: effect of nozzle diameter,** J. Mater. Sci., 16 (12) (2005), pp. 1159-1163, 10.1007/s10856-005-4723-6
16. J.P. Li, J.R. de Wijn, C.A. Van Blitterswijk, K. de Groot, **Porous Ti6Al4V scaffold directly fabricating by rapid prototyping: preparation and in vitro experiment,** Biomaterials, 27 (8) (2006), pp. 1223-1235, 10.1016/j.biomaterials.2005.08.033
17. H. Elsayed, P. Rebesan, G. Giacomello, M. Pasetto, C. Gardin, L. Ferroni, B. Zavan, L. Biasetto, **Direct ink writing of porous titanium (Ti6Al4V) lattice structures,** Mater. Sci. Eng. C, 103 (2019), p. 109794, 10.1016/j.msec.2019.109794
18. H. Elsayed, N. Novak, M. Vesenjak, F. Zanini, S. Carmignato, L. Biasetto, **The effect of strut size on microstructure and compressive strength of porous Ti6Al4V lattices printed via direct ink writing,** Mater. Sci. Eng. A, 787 (2020), p. 139484, 10.1016/j.msea.2020.139484
19. M. Qian, Y.F. Yang, S.D. Luo, H.P. Tang, **12 - Pressureless sintering of titanium and titanium alloys: sintering densification and solute homogenization,** in M. Qian, F.H. (Sam) Froes (Eds.), Titanium Powder Metallurgy, Butterworth-Heinemann, Boston (2015), pp. 201-218, 10.1016/B978-0-12-800054-0.00012-5
20. Y. Li, X.M. Chou, L. Yu, **Dehydrogenation debinding process of MIM titanium alloys by TiH2 powder,** Powder Metall., 49 (3) (2006), pp. 236-239, 10.1179/174329006X95338
21. M. Qian, G. Schaffer, C. Bettles, **Sintering of titanium and its alloys,** Sintering of Advanced Materials, Elsevier (2010), pp. 324-355, 10.1533/9781845699949.3.324
22. Beck, **Effect of Hydrogen on Mechanical Properties of Titanium and Its Alloys,** Technical Report CR - 134796, NASA (1975)
23. M. Gouné, T. Belmonte, J.M. Fiorani, S. Chomer, H. Michel, **Modelling of diffusion–precipitation in nitrided alloyed iron,** Thin Solid Films, 377-378 (2000), pp. 543-549, 10.1016/S0040-6090(00)01305-5
24. M. Roussel, X. Sauvage, M. Perez, D. Magné, A. Hauet, A. Steckmeyer, M. Vermont, T. Chaise, M. Couvrat, **Influence of solidification induced composition gradients on carbide precipitation in FeNiCr heat resistant steels,** Materialia, 4 (2018), pp. 331-339, 10.1016/j.mtla.2018.10.010

25. D. Bardel, M. Fontaine, T. Chaise, M. Perez, D. Nelias, F. Bourlier, J. Garnier, **Integrated modelling of a 6061-T6 weld joint: from microstructure to mechanical properties,** Acta Mater., 117 (2016), pp. 81-90, 10.1016/j.actamat.2016.06.017
26. Q. Fu, E. Saiz, A.P. Tomsia, **Direct ink writing of highly porous and strong glass scaffolds for load-bearing bone defects repair and regeneration,** Acta Biomater., 7 (10) (2011), pp. 3547-3554, 10.1016/j.actbio.2011.06.030
27. E. Feilden, E.G.-T. Blanca, F. Giuliani, E. Saiz, L. Vandeperre, **Robocasting of structural ceramic parts with hydrogel inks,** J. Eur. Ceram. Soc., 36 (10) (2016), pp. 2525-2533, 10.1016/j.jeurceramsoc.2016.03.001
28. E. Feilden, Additive manufacturing of ceramics and ceramic composites via robocasting (2017). PhD thesis, Imperial College London, UK; 10.25560/55940
29. M. Yetna N'Jock, E. Camposilvan, L. Gremillard, E. Maire, D. Fabrègue, D. Chicot, K. Tabalaiev, J. Adrien, **Characterization of 100Cr6 lattice structures produced by robocasting,** Mater. Des., 121 (2017), pp. 345-354, 10.1016/j.matdes.2017.02.066
30. M.F. Ashby, A.G. Evans, N.A. Fleck, L.J. Gibson, J.W. Hutchinson, H.N.G. Wadley, Metal foams: a design guide (2000) 263.
31. J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D.J. White, V. Hartenstein, K. Eliceiri, P.Tomancak, A. Cardona, **Fiji: an open-source platform for biological-image analysis,** Nat. Methods, 9 (7) (2012), pp. 676-682, 10.1038/nmeth.2019
32. M. Perez, M. Dumont, D. Acevedo, **Implementation of the classical nucleation and growth theory for precipitation,** Acta Mater., 56 (2008), pp. 2119-2132, 10.1016/j.actamat.2007.12.050
33. M. Perez, M. Dumont, D. Acevedo, **Corrigendum to "Implementation of the classical nucleation and growth theory for precipitation",** Acta Mater., 57 (2008), p. 1318, 10.1016/j.actamat.2008.11.020
34. R. Wagner, R. Kampmann, P.W. Voorhees, **Homogeneous second-phase precipitation,** Phase Transformations in Materials, John Wiley & Sons, Ltd (2005), pp. 309-407, 10.1002/352760264X.ch5
35. C. Toffolon-Masclet, A. Perron, B. Mazères, S. Dépinoy, C. Desgranges, L. Martinelli, D. Monceau, X. Boulnat, A. Mathevon, M. Perez, **Computational kinetics: application to nuclear materials,** Comprehensive Nuclear Materials, Elsevier (2020), pp. 850-880, 10.1016/B978-0-12-803581-8.11600-5
36. ThermoTech, TTTI3 database, (http://www.thermotech.co.uk/databases.html#Ti-DATA).
37. M. Perez, **Gibbs-Thomson effect in phase transformations,** Scr. Mater., 52 (2005), pp. 709-712, 10.1016/j.scriptamat.2004.12.026
38. Q. Du, M. Perez, W.J. Poole, M. Wells, **Numerical integration of the Gibbs–Thomson equation for multicomponent systems,** Scr. Mater., 66 (419-422) (2012), 10.1016/j.scriptamat.2011.11.019
39. H.P. Van Landeghem, M. Gouné, A. Redjaïmia, **Nitride precipitation in compositionally heterogeneous alloys: nucleation, growth and coarsening during nitriding,** J. Cryst. Growth, 341 (1) (2012), pp. 53-60, 10.1016/j.jcrysgro.2011.12.056
40. R. Dabrowski, **The kinetics of phase transformations during continuous cooling of the Ti6Al4V alloy from the single-phase β-range,** Arch. Metall. Mater., 56 (3) (2011), pp. 703-707, 10.2478/v10172-011-0077-x
41. K.T. Jacob, S. Gupta, **Calciothermic reduction of TiO2: a diagrammatic assessment of the thermodynamic limit of deoxidation,** JOM, 61 (5) (2009), pp. 56-59, 10.1007/s11837-009-0072-0
42. S. Malinov, W. Sha, Z. Guo, C. Tang, A. Long, **Synchrotron X-ray diffraction study of the phase transformations in titanium alloys,** Mater. Charact., 48 (4) (2002), pp. 279-295, 10.1016/S1044-5803(02)00286-3

43. S.-H. Joo, K. Yubuta, H. Kato, **Ordering kinetics of nanoporous FeCo during liquid metal dealloying and the development of nanofacets,** Scr. Mater., 177 (2020), pp. 38-43, 10.1016/j.scriptamat.2019.10.007
44. Y.-J. Kim, H. Chung, S.-J.L. Kang, **Processing and mechanical properties of Ti–6Al–4V/TiC in situ composite fabricated by gas–solid reaction,** Mater. Sci. Eng. A, 333 (1–2) (2002), pp. 343-350, 10.1016/S0921-5093(01)01858-5
45. L. Huang, L. Geng, H. Xu, H. Peng, **In situ TiC particles reinforced Ti6Al4V matrix composite with a network reinforcement architecture,** Mater. Sci. Eng. A, 528 (6) (2011), pp. 2859-2862, 10.1016/j.msea.2010.12.046
46. S. Mereddy, M.J. Bermingham, D. Kent, A. Dehghan-Manshadi, D.H. StJohn, M.S. Dargusch, **Trace carbon addition to refine microstructure and enhance properties of additive-manufactured Ti-6Al-4V,** JOM, 70 (9) (2018), pp. 1670-1676, 10.1007/s11837-018-2994-x
47. C. Badini, G. Ubertalli, D. Puppo, P. Fino, **High temperature behaviour of a Ti-6Al-4V/TiCp composite processed by BE-CIP-HIP method**, J. Mater. Sci, 35 (2000), pp. 3903-3912, 10.1023/A:1004893700762
48. A.A. da Silva, J.F. dos Santos, T.R. Strohaecker, **An investigation of the fracture behaviour of diffusion-bonded Ti6Al4V/TiC/10p,** Compos. Sci. Technol., 66 (13) (2006), pp. 2063-2068, 10.1016/j.compscitech.2005.12.018
49. D. Poquillon, C. Armand, J. Huez, **Oxidation and oxygen diffusion in Ti–6al–4V Alloy: improving measurements during SIMS analysis by rotating the sample,** Oxid. Metals, 79 (3-4) (2013), pp. 249-259, 10.1007/s11085-013-9360-8
50. C. Zener, **Private communication to CS Smith,** Trans. TMS-AIME (1948), p. 175
51. P. Rios, **Overview no-62 - a theory for grain-boundary pinning by particles,** Acta Metall., 35 (12) (1987), pp. 2805-2814, 10.1016/0001-6160(87)90280-X
52. M. Yan, W. Xu, M.S. Dargusch, H.P. Tang, M. Brandt, M. Qian, **Review of effect of oxygen on room temperature ductility of titanium and titanium alloys,** Powder Metall., 57 (4) (2014), pp. 251-257, 10.1179/1743290114Y.0000000108
53. J. Chávez, L. Olmos, O. Jiménez, D. Bouvard, E. Rodríguez, M. Flores, **Sintering behaviour and mechanical characterisation of Ti64/xTiN composites and bilayer components,** Powder Metall., 60 (4) (2017), pp. 257-266, 10.1080/00325899.2017.1280585
54. B. Song, S. Dong, H. Liao, C. Coddet, **Process parameter selection for selective laser melting of Ti6Al4V based on temperature distribution simulation and experimental sintering,** Int. J. Adv. Manuf. Technol., 61 (9-12) (2012), pp. 967-974, 10.1007/s00170-011-3776-6
55. J.-M. Oh, K.-H. Heo, W.-B. Kim, G.-S. Choi, J.-W. Lim, **Sintering properties of Ti-6Al-4V alloys prepared using Ti/TiH powders,** Mater. Trans., 54 (1) (2013), pp. 119-121, 10.2320/matertrans.M2012304
56. A.A. da Silva, J.F. dos Santos, T.R. Strohaecker, **Microstructural and mechanical characterisation of a Ti6Al4V/TiC/10p composite processed by the BE-CHIP method,** Compos. Sci. Technol., 65 (11-12) (2005), pp. 1749-1755, 10.1016/j.compscitech.2005.03.005

## Appendix A. Supplementary materials

In the notebook presented in the supplementary materials section, next page, we'll simulate the diffusion of C and O and the precipitation/dissolution of TiC in Ti of Ti6Al4V alloy during debinding and sintering.

- The notebook is also avalaible here: https://ars-els-cdn-com.docelec.insa-lyon.fr/content/image/1-s2.0-S1359645421006042-mmc1.pdf. This is open data under the CC by licence http://creativecommons.org/licenses/by/4.0/
- See [Perez08] for general description of the model.
- See [PreciSo] to get the PreciSo Pyhton module used here.

# GlobalScenario-3nodes-Precipitation-Iso

March 12, 2021

## 1 Content

In this noteboook, we'll simulate the diffusion of C and O and the precipitation/dissolution of TiC in Ti of Ti6Al4V alloy during debinding and sintering. See **[Perez08]** for general description of the model. See **[PreciSo]** to get the PreciSo Pyhton module used here.

## 2 Imports, colors and constants

[1]:
```
import preciso
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt
import os
from scipy import interpolate
import pandas as pd
from scipy import stats
from glob import glob
import math

colors=['green','red','blue','black','white']
colorsChem=['firebrick','mediumslateblue']
markers = ["o", "^", ".", "x", "v", "<", ">", "d", "s","o", "^", ".", "x", "v",␣
↪"<", ">", "d", "s"]

M_O=15.9994e-3
M_C=12.0107e-3
M_H=1.00794e-3
M_Ti=47.867e-3
```

## 3 Parameters of the simulation

### 3.1 System and precipitates name

[2]:
```
systemName='Ti64'
precipitateName=["TiC"]
soluteElementsName=["C","O"]
```

## 3.2 Solubility limits

The solubility product $\log_{10} X_{\mathrm{Ti}} X_{\mathrm{C}} = \frac{C}{T^2} - \frac{A}{T} + B$, where $X_j$ is in atom fraction. Following values are extracted from ThermoCalc calculations where only TiC and matrix with various O levels are active within TTTI4 database.

### 3.2.1 TiC in $\alpha$ (HCP) and $\beta$ BCC

[3]:
```
phase=["alpha","beta"]
phaseTC=["HCP_A3","BCC_A2"]
ConcentrationO=["0","02","04","06","08","1"]
C_O=np.asarray([0,0.2,0.4,0.6,0.8,1])

A_TiC=[]
B_TiC=[]
C_TiC=[]

fig, ((ax11, ax12),(ax21,ax22),(ax31,ax32)) = plt.subplots(3, 2,figsize=(14,21))

# Solubility product versus temperature
for i in range(len(phase)):
    A_TiC.append([])
    B_TiC.append([])
    C_TiC.append([])
    for j in range(len(ConcentrationO)):
        file='SolubilityProduct_'+phase[i]+'-'+ConcentrationO[j]+'wtpcO.txt'
        soluteContent=pd.read_csv(file,sep='\t',header =0)
        OneOverT=1/(soluteContent['Temperature [°C]']+273.15)
        SolProd=np.log10(soluteContent['Mole fraction of Ti in '+phaseTC[i]]**6
                    *soluteContent['Mole fraction of C in '+phaseTC[i]]**4)␣
↪#Ti6C4

        ax11.scatter(OneOverT, SolProd, color=colors[0], label='label',
            marker=markers[0], s=10)
        slope, intercept, r_value, p_value, std_err = stats.
↪linregress(OneOverT,SolProd)

        z = np.polyfit(OneOverT,SolProd, 2)
        p = np.poly1d(z)

        ax11.plot(OneOverT, slope*OneOverT+intercept, color=colors[1],␣
↪label='label')
        ax11.plot(OneOverT, OneOverT**2*z[0]+OneOverT*z[1]+z[2],␣
↪color=colors[1], label='label')

        A_TiC[i].append(-z[1])
        B_TiC[i].append(z[2])
```

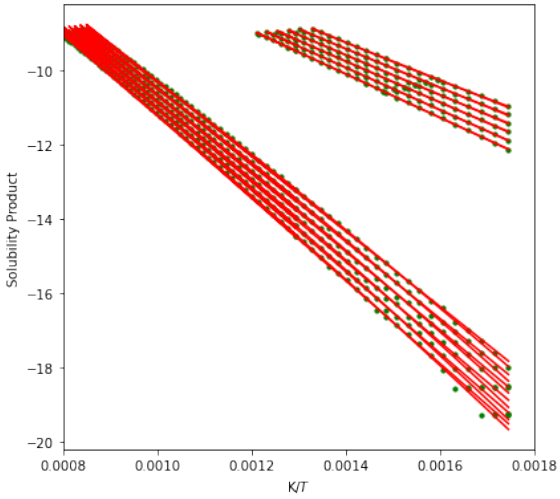
```
        C_TiC[i].append(z[0])
ax11.set_xlim([0.0008,0.0018])
ax11.set_xlabel('K/$T$')
ax11.set_ylabel('Solubility Product')


# A parameter of solubility product vs O concentration
slope_A_TiC=[]
intercept_A_TiC=[]
for i in range(len(phase)):
    ax12.scatter(C_O, A_TiC[i], color=colors[0], label='label',
            marker=markers[0], s=50)
    slope, intercept, r_value, p_value, std_err = stats.linregress(C_O,A_TiC[i])
    slope_A_TiC.append(slope)
    intercept_A_TiC.append(intercept)
    ax12.plot(C_O, slope_A_TiC[i]*np.asarray(C_O)+intercept_A_TiC[i],␣
 ↪color=colors[1], label='label')
for i in range(len(phase)):
    print("slope_A_TiC in",phase[i],"(K/wt%):", slope_A_TiC[i])
    print("intercept_A_TiC in",phase[i],"(K):",intercept_A_TiC[i])
ax12.set_xlabel('O concentration (wt%)')
ax12.set_ylabel('A parameter of solubility Product (K)')

# B parameter of solubility product vs O concentration
slope_B_TiC=[]
intercept_B_TiC=[]
for i in range(len(phase)):
    ax21.scatter(C_O, B_TiC[i], color=colors[0], label='label',
            marker=markers[0], s=50)
    slope, intercept, r_value, p_value, std_err = stats.linregress(C_O,B_TiC[i])
    slope_B_TiC.append(slope)
    intercept_B_TiC.append(intercept)
    ax21.plot(C_O, slope_B_TiC[i]*np.asarray(C_O)+intercept_B_TiC[i],␣
 ↪color=colors[1], label='label')
for i in range(len(phase)):
    print("slope_B_TiC in",phase[i],"(1/wt%):", slope_B_TiC[i])
    print("intercept_B_TiC in",phase[i],":",intercept_B_TiC[i])

ax21.set_xlabel('O concentration (wt%)')
ax21.set_ylabel('B parameter of solubility Product (K)')

# C parameter of solubility product vs O concentration
slope_C_TiC=[]
intercept_C_TiC=[]
for i in range(len(phase)):
    ax22.scatter(C_O, C_TiC[i], color=colors[0], label='label',
            marker=markers[0], s=50)
```

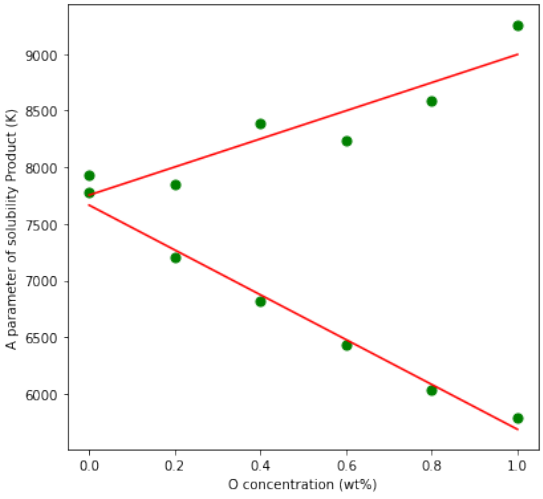
```
    slope, intercept, r_value, p_value, std_err = stats.linregress(C_O,C_TiC[i])
    slope_C_TiC.append(slope)
    intercept_C_TiC.append(intercept)
    ax22.plot(C_O, slope_C_TiC[i]*np.asarray(C_O)+intercept_C_TiC[i],␣
 ↪color=colors[1], label='label')
for i in range(len(phase)):
    print("slope_C_TiC in",phase[i],"(K^2/wt%):", slope_C_TiC[i])
    print("intercept_C_TiC in",phase[i],"(K^2):",intercept_C_TiC[i])

ax22.set_xlabel('O concentration (wt%)')
ax22.set_ylabel('C parameter of solubility Product (K)')

# Solubility product vs temperature with A, B and C
for i in range(len(phase)):
    for j in range(len(ConcentrationO)):
        file='SolubilityProduct_'+phase[i]+'-'+ConcentrationO[j]+'wtpcO.txt'
        soluteContent=pd.read_csv(file,sep='\t',header =0)
        OneOverT=1/(soluteContent['Temperature [°C]']+273.15)
        SolProd=np.log10(soluteContent['Mole fraction of Ti in '+phaseTC[i]]**6
                    *soluteContent['Mole fraction of C in '+phaseTC[i]]**4)

        ax31.scatter(OneOverT, SolProd, color=colors[0], label='label',
            marker=markers[0], s=10)
        A=slope_A_TiC[i]*C_O[j]+intercept_A_TiC[i]
        B=slope_B_TiC[i]*C_O[j]+intercept_B_TiC[i]
        C=slope_C_TiC[i]*C_O[j]+intercept_C_TiC[i]

        ax31.plot(OneOverT, OneOverT**2*C-OneOverT*A+B, color=colors[1],␣
 ↪label='label')
ax31.set_xlim([0.0008,0.0018])
ax31.set_xlabel('K/$T$')
ax31.set_ylabel('Solubility Product')

# alpha <=> beta transition temperature
T_trans_TCC=np.array([923,942,969,1000,1020,1050]) # calculation for Ti6Al4V0.2C
ax32.scatter(C_O, T_trans_TCC, color=colors[0], label='label',
            marker=markers[0], s=50)
slope, intercept, r_value, p_value, std_err = stats.linregress(C_O,T_trans_TCC)
slope_T_trans_O=slope
intercept_T_trans_O=intercept
ax32.plot(C_O, slope_T_trans_O*np.asarray(C_O)+intercept_T_trans_O,␣
 ↪color=colors[1], label='label')
ax32.set_xlabel('O concentration (wt%)')
ax32.set_ylabel('alpha $\leftrightarrow$ beta transition temperature (°C)')
```

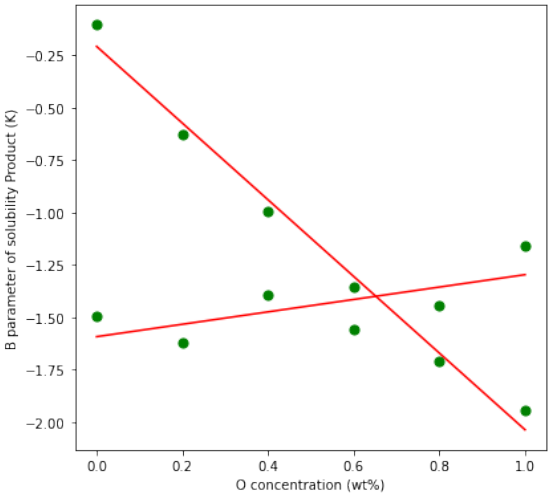
```
slope_A_TiC in alpha (K/wt%): -1978.2503930490223
intercept_A_TiC in alpha (K): 7665.317074250619
```

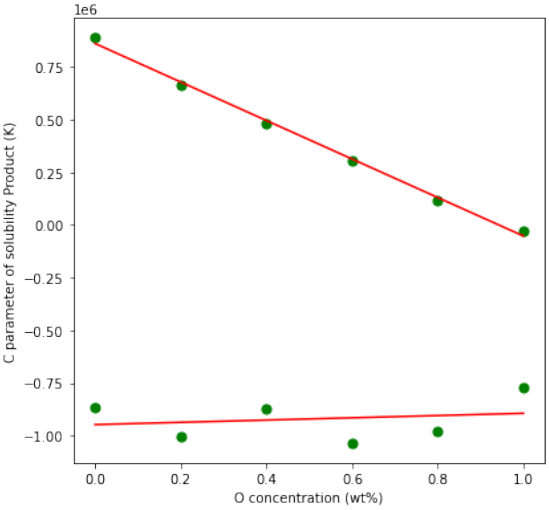
```
slope_A_TiC in beta (K/wt%): 1241.026014282342
intercept_A_TiC in beta (K): 7753.832623342463
slope_B_TiC in alpha (1/wt%): -1.8266685144131223
intercept_B_TiC in alpha : -0.20972323965909367
slope_B_TiC in beta (1/wt%): 0.2953905487407387
intercept_B_TiC in beta : -1.5928359319362277
slope_C_TiC in alpha (K^2/wt%): -913277.4937344381
intercept_C_TiC in alpha (K^2): 859620.7344221715
slope_C_TiC in beta (K^2/wt%): 53624.09532183275
intercept_C_TiC in beta (K^2): -947862.7261623928
```

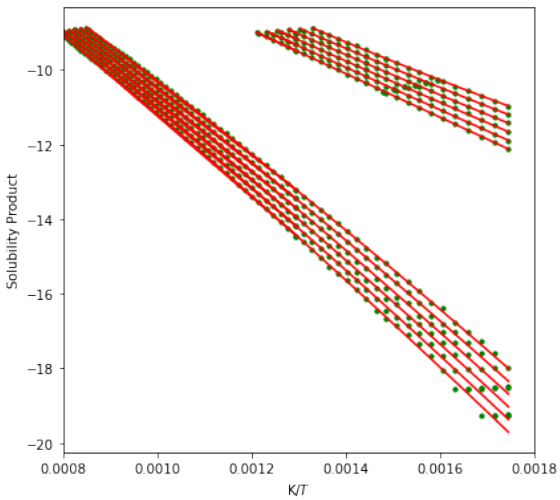
[3]: `Text(0, 0.5, 'alpha $\\leftrightarrow$ beta transition temperature (°C)')`

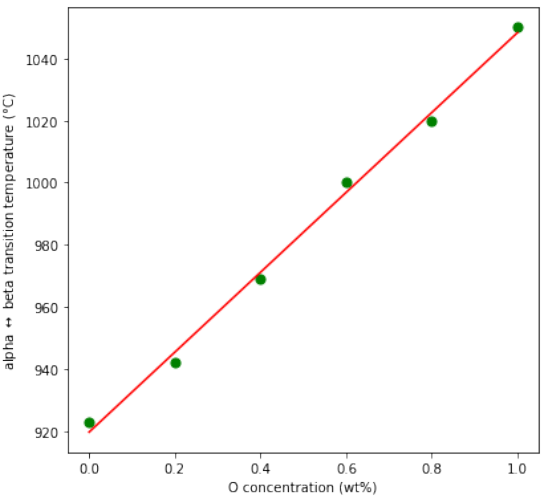

Solubility Product
K/T


A parameter of solubility Product (K)
O concentration (wt%)


B parameter of solubility Product (K)
O concentration (wt%)


1e6
C parameter of solubility Product (K)
O concentration (wt%)


Solubility Product
K/T


alpha ↔ beta transition temperature (°C)
O concentration (wt%)

### 3.2.2 Solubility limits calculator

[4]:
```
Conc_O=0.1
i=0 # alpha
#i=1 # beta
A=slope_A_TiC[i]*Conc_O+intercept_A_TiC[i]
B=slope_B_TiC[i]*Conc_O+intercept_B_TiC[i]
C=slope_C_TiC[i]*Conc_O+intercept_C_TiC[i]
Temp=500 #°C
SolLimit_atfrac=(10**((1/(273+Temp))**2*C-(1/(273+Temp))*A+B))**(1/4)
SolLimit_wtpc=M_C*SolLimit_atfrac/
 ↪(M_C*SolLimit_atfrac+M_Ti*(1-SolLimit_atfrac))*100
print("Solubility limit :",SolLimit_wtpc)
```

```
Solubility limit : 0.16213494195798864
```

## 3.3 Diffusion coefficients

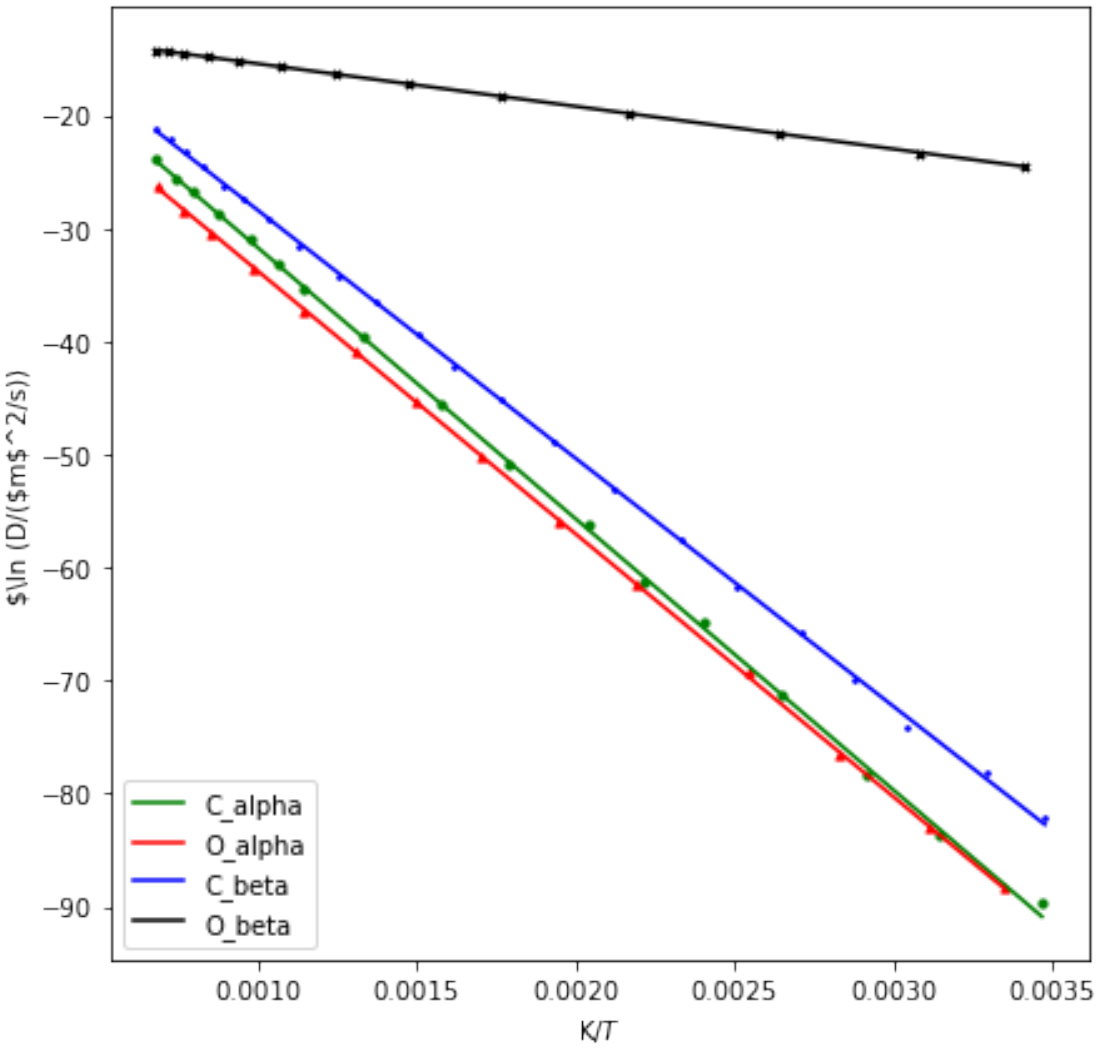
### 3.3.1 Values form TTTI3 mobiliti database

[5]:
```
fig, ax = plt.subplots(figsize=(7,7))
D0_TCC = [[0 for x in range(len(phase))] for y in␣
 ↪range(len(soluteElementsName))]
Q_TCC = [[0 for x in range(len(phase))] for y in range(len(soluteElementsName))]
col=0
for p in range(len(phase)):
    for s in range(len(soluteElementsName)):
        file='Diffusion_'+soluteElementsName[s]+'_'+phase[p]+'_TCC.csv'
        DiffusionData=pd.read_csv(file,sep='\t',header =0)

        OneOverT=1/(DiffusionData['T [°C]']+273.15)
        ln_D=np.log(DiffusionData['D [m^2/s]'])

        label=soluteElementsName[s]+'_'+phase[p]

        ax.scatter(OneOverT, ln_D, color=colors[col], label="",␣
 ↪marker=markers[col], s=10)
        slope, intercept, r_value, p_value, std_err = stats.
 ↪linregress(OneOverT,ln_D)
        D0_TCC[p][s]=np.exp(intercept)
        Q_TCC[p][s]=-slope*8.31
        print("D_0 for "+soluteElementsName[s]+" in "+phase[p]+" :
 ↪",D0_TCC[p][s], "m$^2$/s")
        print("Q "+soluteElementsName[s]+" in "+phase[p]+" :",Q_TCC[p][s],"J/
 ↪mol")
        ax.plot(OneOverT, slope*OneOverT+intercept, color=colors[col],␣
 ↪label=label)
        col+=1
```

```
ax.set_xlabel('K/$T$')
ax.set_ylabel('$\ln (D/($m$^2/s))')
ax.legend()
```

```
D_O for C in alpha : 0.00040822748788177843 m$^2$/s
Q C in alpha : 199026.8237747762 J/mol
D_O for O in alpha : 2.497920324704343e-05 m$^2$/s
Q O in alpha : 192974.9625802615 J/mol
D_O for C in beta : 0.0014644083601068052 m$^2$/s
Q C in beta : 182096.67758744897 J/mol
D_O for O in beta : 8.375495183268161e-06 m$^2$/s
Q O in beta : 31298.249802102302 J/mol
```

[5]: <matplotlib.legend.Legend at 0x7ff774afc310>



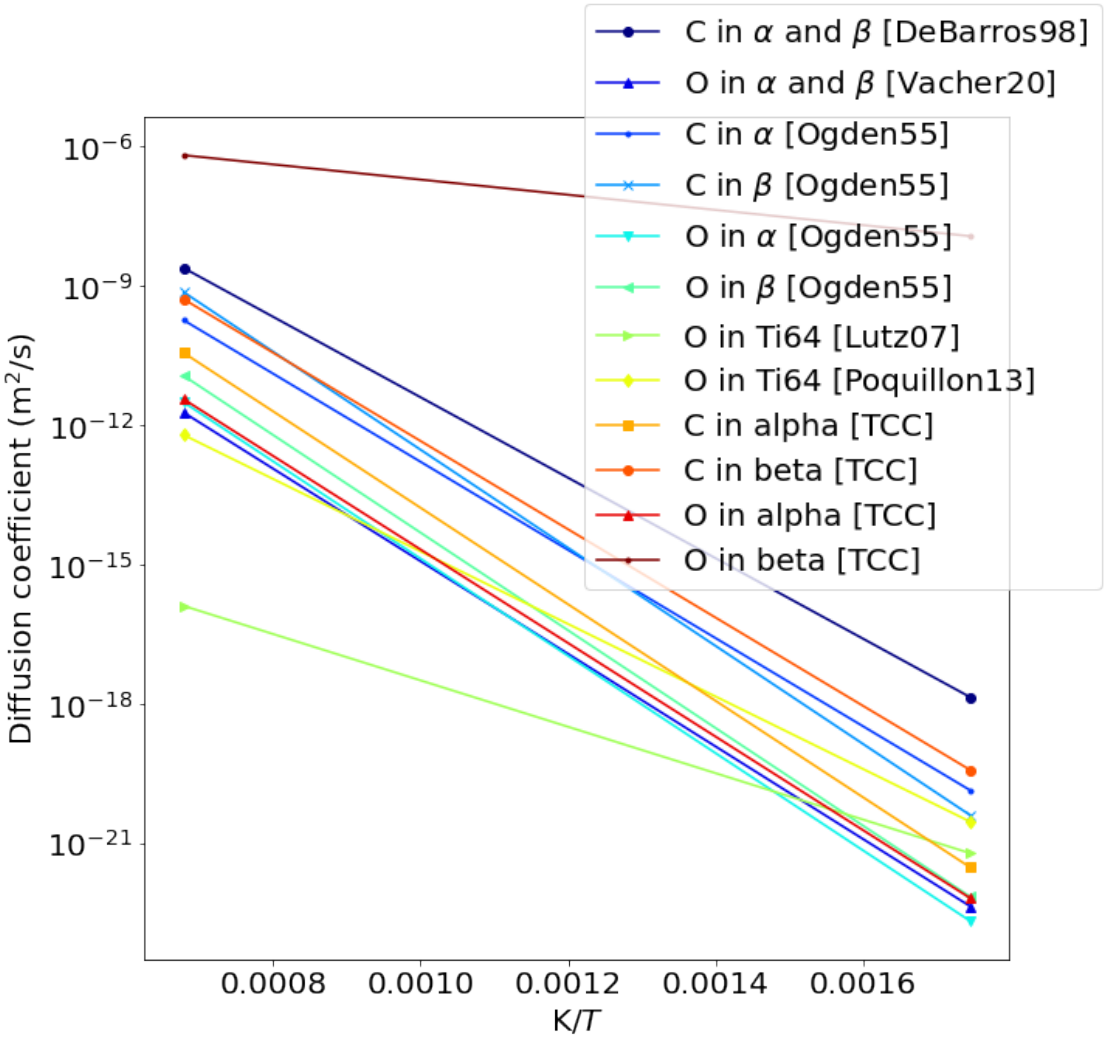
### 3.3.2 Values from various sources of the litterature

[6]:
```
import matplotlib.pylab as pl

D_author=[]
D0_lit=[]
Q_lit=[]

# 0
D_author.append("C in $\\alpha$ and $\\beta$ [DeBarros98]")
D0_lit.append(18.5e-4)
Q_lit.append(166e3)

# 1
D_author.append("O in $\\alpha$ and $\\beta$ [Vacher20]")
D0_lit.append(1.1e-5)
Q_lit.append(191e3)

# 2
D_author.append("C in $\\alpha$ [Ogden55]")
D0_lit.append(5.06e-4)
Q_lit.append(43500*4.18)


# 3
D_author.append("C in $\\beta$ [Ogden55]")
D0_lit.append(108e-4)
Q_lit.append(48400*4.18)

# 4
D_author.append("O in $\\alpha$ [Ogden55]")
D0_lit.append(0.4e-4)
Q_lit.append(48000*4.18)

# 5
D_author.append("O in $\\beta$ [Ogden55]")
D0_lit.append(1.6e-4)
Q_lit.append(48200*4.18)

# 6
D_author.append("O in Ti64 [Lutz07]")
T1_l=1000/0.9
T2_l=1000/1.7
D1_l=1e-13/1e4
D2_l=1e-17/1e4
```

```
Q_Lutz=-8.31*np.log(D1_l/D2_l)/(1/T1_l-1/T2_l)
D0_Lutz=D1_l/np.exp(-Q_Lutz/8.31/T1_l)
D0_lit.append(D0_Lutz)
Q_lit.append(Q_Lutz)

# 7
D_author.append("O in Ti64 [Poquillon13]")
T1_p=1/0.001
T2_p=1/0.0012
D1_p=np.exp(-20-47/50*5)/10000
D2_p=np.exp(-20-83/50*5)/10000
Q_Poquillon=-8.31*np.log(D1_p/D2_p)/(1/T1_p-1/T2_p)
D0_Poquillon=D1_p/np.exp(-Q_Poquillon/8.31/T1_p)
D0_lit.append(D0_Poquillon)
Q_lit.append(Q_Poquillon)

# 8
D_author.append("C in alpha [TCC]")
D0_C_alpha_TCC=D0_TCC[0][0]
Q_C_alpha_TCC=Q_TCC[0][0]
D0_lit.append(D0_C_alpha_TCC)
Q_lit.append(Q_C_alpha_TCC)

# 9
D_author.append("C in beta [TCC]")
D0_C_beta_TCC=D0_TCC[1][0]
Q_C_beta_TCC=Q_TCC[1][0]
D0_lit.append(D0_C_beta_TCC)
Q_lit.append(Q_C_beta_TCC)

# 10
D_author.append("O in alpha [TCC]")
D0_O_alpha_TCC=D0_TCC[0][1]
Q_O_alpha_TCC=Q_TCC[0][1]
D0_lit.append(D0_O_alpha_TCC)
Q_lit.append(Q_O_alpha_TCC)

# 11
D_author.append("O in beta [TCC]")
D0_O_beta_TCC=D0_TCC[1][1]
Q_O_beta_TCC=Q_TCC[1][1]
D0_lit.append(D0_O_beta_TCC)
Q_lit.append(Q_O_beta_TCC)

cols = pl.cm.jet(np.linspace(0,1,len(D_author)))
OneOverT_Diff=np.asarray([1/(273+300),1/(273+1200)])
fig,ax =plt.subplots(figsize=(10,10))
```

```
for i in range(len(D_author)):
    D=D0_lit[i]*np.exp(-Q_lit[i]/8.32*OneOverT_Diff)
    plt.semilogy(OneOverT_Diff,D, color=cols[i], label=D_author[i],␣
 ↪marker=markers[i])

fig.legend(fontsize=20)
ax.set_xlabel('K/$T$',fontsize=20)
ax.set_ylabel('Diffusion coefficient (m$^2$/s)',fontsize=20)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
```

[6]:
```
(array([1.e-27, 1.e-24, 1.e-21, 1.e-18, 1.e-15, 1.e-12, 1.e-09, 1.e-06,
        1.e-03, 1.e+00]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```




### 3.3.3 Diffusion length

[7]:
```
n_model_diff=5
T=600 #°C
t=600 #s
print('Diffusion length (micron): ',np.sqrt(D0_lit[n_model_diff]*np.
 ↪exp(-Q_lit[n_model_diff]/8.31/(T+273))*t)*1e6)
print('Element and author: ',D_author[n_model_diff])
print('Activation energy (kJ): ',Q_lit[n_model_diff]/1000)
print('Pre-exp. factor (m^2/s): ',D0_lit[n_model_diff])
```

```
Diffusion length (micron):  0.2887420262268569
Element and author:  O in $\beta$ [Ogden55]
Activation energy (kJ):  201.476
Pre-exp. factor (m^2/s):  0.00016
```

### 3.3.4 Chosen values from the litterature

[8]:
```
i_C_alpha=2
D0_C_alpha=D0_lit[i_C_alpha]
Q_C_alpha=Q_lit[i_C_alpha]
print("C in alpha:",D_author[i_C_alpha])
print("D0_C_alpha:",D0_C_alpha)
print("Q_C_alpha:",Q_C_alpha)

i_C_beta=3
D0_C_beta=D0_lit[i_C_beta]
Q_C_beta=Q_lit[i_C_beta]
print("C in beta:",D_author[i_C_beta])
print("D0_C_beta:",D0_C_beta)
print("Q_C_beta:",Q_C_beta)

i_O_alpha=7
D0_O_alpha=D0_lit[i_O_alpha]
Q_O_alpha=Q_lit[i_O_alpha]
print("O in alpha:",D_author[i_O_alpha])
print("D0_O_alpha:",D0_O_alpha)
print("Q_O_alpha:",Q_O_alpha)

i_O_beta=5
D0_O_beta=D0_lit[i_O_beta]
Q_O_beta=Q_lit[i_O_beta]
print("O in beta:",D_author[i_O_beta])
print("D0_O_beta:",D0_O_beta)
print("Q_O_beta:",Q_O_beta)
```

```
C in alpha: C in $\alpha$ [Ogden55]
D0_C_alpha: 0.000506
Q_C_alpha: 181830.0
C in beta: C in $\beta$ [Ogden55]
D0_C_beta: 0.0108
Q_C_beta: 202312.0
O in alpha: O in Ti64 [Poquillon13]
D0_O_alpha: 1.2309119026734822e-07
Q_O_alpha: 149580.0
O in beta: O in $\beta$ [Ogden55]
D0_O_beta: 0.00016
Q_O_beta: 201476.0
```

## 3.4 Surface energy (in $J/m^2$)

Fitting parameter of the model

[9]:
```
gamma_alpha=0.2
gamma_beta=0.2
```

### 3.5 Local carbon and oxygen content in the C/O rich layer

[10]:
```
binder_C_C_wtpc=2
binder_C_O_wtpc=15
```

### 3.6 Lattice parameters and atomic volumes

[11]:
```
a_alpha=0.2935e-9
c_alpha=0.4673e-9
a_beta=0.3226e-9
a_TiC=4.35e-10
natom_per_latt_alpha=2
natom_per_latt_beta=2
natom_per_latt_TiC=8
v_at_alpha=a_alpha**2*math.sqrt(3)/2*c_alpha/natom_per_latt_alpha
v_at_beta=a_beta**3/natom_per_latt_beta
v_at_TiC=a_TiC**3/natom_per_latt_TiC
print("Atomic volume of alpha (m^3):",v_at_alpha)
print("Atomic volume of beta (m^3):",v_at_beta)
print("Atomic volume of TiC (m^3):",v_at_TiC)
```

```
Atomic volume of alpha (m^3): 1.7430611698467416e-29
Atomic volume of beta (m^3): 1.6786613588e-29
Atomic volume of TiC (m^3): 1.0289109375e-29
```

#### 3.6.1 Temperature (in °C) and furnace parameters

[12]:
```
TK=273             # Temperature in K
RT=30              # Room temperature

T0=1200            # Caracteristic cooling temperature
n_furnace=0.7      # Exponent for cooling
tau_furnace=150*60   # Caracteristic time for cooling
DeltaT_cool=50   # Cooling ramp

T_deb=[350,500]    # Debinding temperatures
t_deb=[1800,7200] # Debinding times
T_sin=[1200,1200] # Sintering temperatures
t_sin=[7200,7200] # Sintering times

T_end_deb=555   # Temperature at which debinding layer is gone

HR_deb=1/60 # Heating rate for debinding
```

```
HR_sin=10/60 # Heating rate for sintering
```

### 3.6.2 Geometry of the system

[13]:
```
part_rad=12.5e-6 # node 0+2
layer_th=1e-6 # node 2
binder_th=1e-6 # node 1
Surf_Nodes02=4*math.pi*(part_rad-layer_th)**2
Surf_Nodes12=4*math.pi*part_rad**2
Vol_Node0=4/3*math.pi*(part_rad-layer_th)**3
Vol_Node2=4/3*math.pi*part_rad**3-Vol_Node0
Vol_Node1=4/3*math.pi*((part_rad+binder_th)**3-part_rad**3)
Pos_Node1=part_rad+layer_th+binder_th/2
Pos_Node2=part_rad+layer_th/2
```

# 4 Template input file for PreciSo

Let's write down our PreciSo configuration file (hereafter called an Input File). See how it's formatted in the **wiki page of PreciSo**.

[14]:
```
input_template =␣
 ↪"""#############################################################################
#                            {{systemName}} - PreciSo input file                ␣
 ↪   #
#############################################################################

# Number of nodes
nodes {{nnodes}}

# Matrix name latticeParameter[m] atomicVolume[m3] molarMass[Kg/mol]
matrix Ti {{lattice_parameter}} {{atomic_volume}} 47.867e-3

# "chimistryArray" is writen is the order of elements
# Element name content[wt_pct] molarMass[Kg/mol] diffusion_D0[m2/s]␣
 ↪diffusion_Q[J/mol]
element C {{C_Cwtpc_binder}} 12.0107e-3 {{D0_C}} {{Q_C}}
element O {{C_Owtpc_binder}} 15.9994e-3 {{D0_O}} {{Q_O}}

# Precipitate name atomiqueVolume[m3] surfaceEnergy[J/m2] solubilityProduct_A␣
 ↪solubilityProduct_B precipitateShapes (aspectRatio if not sphere)␣
 ↪nucleationtype(+ nothing if homogeneous, +heterogeneousNucleationCoeffcient,␣
 ↪number of nucleation sites if heterogeneous) Element1 ChemistryCoefficient1␣
 ↪Element2 ChemistryCoefficient2 Element3 ChemistryCoefficient3...
precipitate TiC 1.0289109375e-29 {{gamma}} {{A}} {{B}} {{C}} sphere homogeneous␣
 ↪Ti 6 C 4
```

```
savethermodynamics results-{{systemName}}-{{TinC}} 1
#classManagementType distrib
#classManagementType old
#classManagementType oldWithLess
#classManagementType lin
#classManagementType no
targetClassNumber 500
savedistribution distribution-{{systemName}}-{{TinC}} 1000
temperatureProfile {{profile}}
initialDT 0.001
increaseDT 1.005
reduceDT 0.5
#timeStepManagement 1
#coeffCFLcondition 10


minDissolutionLimit 1e-10
maxDissolutionLimit 2e-10
criterion Rstar
#criterion solContent

maxCriterionIncrease 0.005

# Specific information on node #i (node_index x_node y_node z_node node_volume)
node 0 0 0 0 {{Vol_Node0}}
node 1 {{Pos_Node1}} 0 0 {{Vol_Node1}}
node 2 {{Pos_Node2}} 0 0 {{Vol_Node2}}

# Connectivity table (node#i node#j surface_bet_#i_and_#j)
connect 0 2 {{Surf_Nodes02}}


# change solute content of nodes 0 and 2
nodeProperty 0 0 element C {{C_Cwtpc_bulk}} 12.0107e-3 {{D0_C}} {{Q_C}}
nodeProperty 0 0 element O {{C_Owtpc_bulk}} 15.9994e-3 {{D0_O}} {{Q_O}}
nodeProperty 2 2 element C {{C_Cwtpc_surf}} 12.0107e-3 {{D0_C}} {{Q_C}}
nodeProperty 2 2 element O {{C_Owtpc_surf}} 15.9994e-3 {{D0_O}} {{Q_O}}

# no precipitation on nodes 1 and 2
nodeProperty 1 1 noPrecipitation
nodeProperty 2 2 noPrecipitation

# Uncomment the next line to study only diffusion
# nodeProperty 0 0 noPrecipitation
```

```
{{OptionalCommand}}

"""
```

#### 4.0.1 fill template with generic values

[15]:
```
generic_values = {
    "Surf_Nodes02":Surf_Nodes02,
    "Surf_Nodes12":Surf_Nodes12,
    "Vol_Node0":Vol_Node0,
    "Vol_Node1":Vol_Node1,
    "Vol_Node2":Vol_Node2,
    "Pos_Node1":Pos_Node1,
    "Pos_Node2":Pos_Node2,
    }
```

[16]:
```
results=[]
stages=[]
```

# 5 Debinding in $\alpha$ (stage 0)

## 5.1 Temperature profiles (in °C)

[17]:
```
T_profile=[]

for i in range(len(T_deb)):
    t_h=(T_deb[i]-RT)/HR_deb
    profile="0.01 "+str(RT+TK)+" "+str(t_h)+" "+str(T_deb[i]+TK)+"␣
 ↪"+str(t_h+t_deb[i])+" "+str(T_deb[i]+TK)
    t_cool=t_h+t_deb[i]
    T_cool=T_deb[i]
    ti=tau_furnace*(math.log(T0/T_cool))**(1/n_furnace)
    T_cool=T_deb[i]-DeltaT_cool
    while(T_cool>RT):
        deltat_cool=tau_furnace*(math.log(T0/T_cool))**(1/n_furnace) -ti
        profile = profile+" "+str(t_cool+deltat_cool)+" "+str(T_cool+TK)
        T_cool=T_cool-DeltaT_cool
    deltat_cool=tau_furnace*(math.log(T0/RT))**(1/n_furnace) -ti
    profile = profile+" "+str(t_cool+deltat_cool)+" "+str(RT+TK)
    profile = profile+" "+str(t_cool+deltat_cool+(T_end_deb-RT)/HR_sin)+"␣
 ↪"+str(T_end_deb+TK)
    T_profile.append(profile)
#print(T_profile)
```

### 5.2 Fill the template file and run PreciSo

[18]:
```
condition="debinding"
nstage=0
results.append([])
stages.append(condition)
print("Stage "+str(nstage)+": "+condition)

nnodes=3
OptionalCommand="""connect 1 2 """+str(Surf_Nodes12)+"""
classManagementType distrib"""

# solubility product in alpha => index 0
local_C_O_wtpc=0.1
A=slope_A_TiC[0]*local_C_O_wtpc+intercept_A_TiC[0]
B=slope_B_TiC[0]*local_C_O_wtpc+intercept_B_TiC[0]
C=slope_C_TiC[0]*local_C_O_wtpc+intercept_C_TiC[0]

for i in range(len(T_profile)):
        # Key - Values to be inserted in the template
        values = {"systemName":systemName+"-"+condition,
          "nnodes":nnodes,
          "profile":T_profile[i],
          "C_Cwtpc_binder":binder_C_C_wtpc,
          "C_Owtpc_binder":binder_C_O_wtpc,
          "C_Cwtpc_bulk":0.011,
          "C_Owtpc_bulk":local_C_O_wtpc,
          "C_Cwtpc_surf":0.011,
          "C_Owtpc_surf":local_C_O_wtpc,
          "TinC":T_deb[i],
          "D0_C":D0_C_alpha,
          "Q_C":Q_C_alpha,
          "D0_O":D0_O_alpha,
          "Q_O":Q_O_alpha,
          "gamma":gamma_alpha,
          "lattice_parameter":a_alpha,
          "atomic_volume":v_at_alpha,
          "A":A,
          "B":B,
          "C":C,
          "OptionalCommand":OptionalCommand}
        values.update(generic_values)

        # fill the template
        input_file = preciso.fillTemplate(input_template, values)

        # save the input file
```

```
        input_fileName=Path(systemName+'-'+condition+'-'+str(T_deb[i])+'.input')
        print("Input file name: "+str(input_fileName))
        with open(input_fileName, 'w') as f:
            f.write(input_file)
        print(input_fileName)
        # run PreciSo
        results[nstage].append(preciso.runSimulation(input_fileName, debug=␣
↪False,temp=False))
```

```
Stage 0: debinding
Input file name: Ti64-debinding-350.input
Ti64-debinding-350.input
Input file name: Ti64-debinding-500.input
Ti64-debinding-500.input
```

### 5.2.1 Solute C and O contents after debinding in both layers

[19]:
```
# C_C_final_wtpc[node][debinding_condition]
C_C_final_wtpc=[]
C_O_final_wtpc=[]

for i in range(nnodes):
    C_C_final_wtpc.append([])
    C_O_final_wtpc.append([])
    for j in range(len(T_deb)):
        data=results[0][j].precipitation[i]

        XC=data["X_C"].iloc[-1]
        XO=data["X_O"].iloc[-1]
        C_C_final_wtpc[i].append(XC*M_C/(XC*M_C+XO*M_O+(1-XC-XO)*M_Ti)*100)
        C_O_final_wtpc[i].append(XO*M_O/(XC*M_C+XO*M_O+(1-XC-XO)*M_Ti)*100)
        #print(XC,XO)
print(C_C_final_wtpc,C_O_final_wtpc)
#print(results[0][i].precipitation[1])
```

```
[[0.011826369641570043, 0.04510133987349704], [1.6231116537337682,
1.0189637718894133], [0.3563780028863845, 0.790557137485433]]
[[0.10000694275836199, 0.10118580012358197], [14.90791033317861,
13.524064326364819], [0.20506748932181476, 1.2790079220437314]]
```

# 6 Sintering

[20]:
```
nnodes=3
```

## 6.1 Heating in $\alpha$ from RT to $T_{trans}$ (stage 1)

### 6.1.1 Fill the template file and run PreciSo

[21]:
```
condition="sintering-heat-alpha"
nstage=1
results.append([])
stages.append(condition)
print("Stage "+str(nstage)+": "+condition)


T_trans=[]


results.append([])
for i in range(len(T_deb)):
    # Transition temperature between alpha and beta
    T_trans.append(slope_T_trans_O*C_O_final_wtpc[0][i]+intercept_T_trans_O)

    t_h=(T_trans[i]-T_end_deb)/HR_sin
    profile="0.01 "+ str(T_end_deb+TK)+" "+str(t_h)+" "+str(T_trans[i]+TK)
    print(profile)

    A=slope_A_TiC[0]*C_O_final_wtpc[0][i]+intercept_A_TiC[0]
    B=slope_B_TiC[0]*C_O_final_wtpc[0][i]+intercept_B_TiC[0]
    C=slope_C_TiC[0]*C_O_final_wtpc[0][i]+intercept_C_TiC[0]

    OptionalCommand="""
    initialDistrib TiC␣
 ↪distribution-Ti64-"""+stages[nstage-1]+"""-"""+str(T_deb[i])+"""TiC_0.dat
    classManagementType distrib"""

    # Key - Values to be inserted in the template
    values = {"systemName":systemName+"-"+condition,
          "nnodes":nnodes,
          "profile":profile,
          "C_Cwtpc_binder":C_C_final_wtpc[0][i], # C from binder is gone after␣
 ↪debinding
          "C_Owtpc_binder":0,
          "C_Cwtpc_bulk":C_C_final_wtpc[0][i],
          "C_Owtpc_bulk":C_O_final_wtpc[0][i],
          "C_Cwtpc_surf":C_C_final_wtpc[2][i],
          "C_Owtpc_surf":C_O_final_wtpc[2][i],
          "TinC":T_deb[i],
          "D0_C":D0_C_alpha,
          "Q_C":Q_C_alpha,
          "D0_O":D0_O_alpha,
          "Q_O":Q_O_alpha,
```

```
            "gamma":gamma_alpha,
            "lattice_parameter":a_alpha,
            "atomic_volume":v_at_alpha,
            "A":A,
            "B":B,
            "C":C,
            "OptionalCommand":OptionalCommand}
    values.update(generic_values)

    # fill the template
    input_file = preciso.fillTemplate(input_template, values)

    # save the input file
    input_fileName=Path(systemName+'-'+condition+'-'+str(T_deb[i])+'.input')
    print("Input file name: "+str(input_fileName))
    with open(input_fileName, 'w') as f:
        f.write(input_file)
    print(input_fileName)
    # run PreciSo
    results[nstage].append(preciso.runSimulation(input_fileName, debug=␣
↪False,temp=False))

#results[1][1].precipitation[0]
```

```
Stage 1: sintering-heat-alpha
0.01 828 2265.433927270736 1205.5723212117894
Input file name: Ti64-sintering-heat-alpha-350.input
Ti64-sintering-heat-alpha-350.input
0.01 828 2266.3433315239063 1205.7238885873176
Input file name: Ti64-sintering-heat-alpha-500.input
Ti64-sintering-heat-alpha-500.input
```

### 6.1.2 Solute C and O contents after heating in both layers

[22]:
```
# C_C_final_wtpc[node][debinding_condition]
C_C_final_wtpc=[]
C_O_final_wtpc=[]

for i in range(nnodes):
    C_C_final_wtpc.append([])
    C_O_final_wtpc.append([])
    for j in range(len(T_deb)):
        data=results[1][j].precipitation[i]

        XC=data["X_C"].iloc[-1]
        XO=data["X_O"].iloc[-1]
        C_C_final_wtpc[i].append(XC*M_C/(XC*M_C+XO*M_O+(1-XC-XO)*M_Ti)*100)
```

```
        C_O_final_wtpc[i].append(XO*M_O/(XC*M_C+XO*M_O+(1-XC-XO)*M_Ti)*100)
        #print(XC,XO)
#print(C_C_final_wtpc,C_O_final_wtpc)
#print(C_O_final_wtpc[0])
```

## 6.2 Heating in $\beta$ from $T_{trans}$ to $T_{sin}$, sinter at $T_{sin}$ and cool down to $T_{trans}$ (stage 2 )

[23]:
```
T_profile=[]

for i in range(len(T_deb)):
    t_h=(T_sin[i]-T_trans[i])/HR_sin
    profile="0.01 "+str(T_trans[i]+TK)+" "+str(t_h)+" "+str(T_sin[i]+TK)+"␣
 ↪"+str(t_h+t_sin[i])+" "+str(T_sin[i]+TK)
    T_profile.append(profile)
#print(T_profile)
```

### 6.2.1 Fill the template file and run PreciSo

[24]:
```
results.append([])
condition="sintering-beta"
nstage=2
stages.append(condition)
print("Stage "+str(nstage)+": "+condition)



for i in range(len(T_deb)):
    A=slope_A_TiC[1]*C_O_final_wtpc[0][i]+intercept_A_TiC[1]
    B=slope_B_TiC[1]*C_O_final_wtpc[0][i]+intercept_B_TiC[1]
    C=slope_C_TiC[1]*C_O_final_wtpc[0][i]+intercept_C_TiC[1]

    OptionalCommand="""
    initialDistrib TiC␣
 ↪distribution-Ti64-"""+stages[nstage-1]+"""-"""+str(T_deb[i])+"""TiC_0.dat
    connect 1 2 """+str(Surf_Nodes12)+"""
    classManagementType distrib"""

    # Key - Values to be inserted in the template
    values = {"systemName":systemName+"-"+condition,
          "nnodes":nnodes,
          "profile":T_profile[i],
          "C_Cwtpc_binder":C_C_final_wtpc[2][i],
          "C_Owtpc_binder":0.8,
          "C_Cwtpc_bulk":C_C_final_wtpc[0][i],
          "C_Owtpc_bulk":C_O_final_wtpc[0][i],
```

```
            "C_Cwtpc_surf":C_C_final_wtpc[2][i],
            "C_Owtpc_surf":C_O_final_wtpc[2][i],
            "TinC":T_deb[i],
            "DO_C":DO_C_beta,
            "Q_C":Q_C_beta,
            "DO_O":DO_O_beta,
            "Q_O":Q_O_beta,
            "gamma":gamma_beta,
            "lattice_parameter":a_beta,
            "atomic_volume":v_at_beta,
            "A":A,
            "B":B,
            "C":C,
            "OptionalCommand":OptionalCommand
        }
    values.update(generic_values)

    # fill the template
    input_file = preciso.fillTemplate(input_template, values)

    # save the input file
    input_fileName=Path(systemName+'-'+condition+'-'+str(T_deb[i])+'.input')
    print("Input file name: "+str(input_fileName))
    with open(input_fileName, 'w') as f:
        f.write(input_file)
    print(input_fileName)
    # run PreciSo
    results[nstage].append(preciso.runSimulation(input_fileName, debug=␣
↪False,temp=False))
```

```
Stage 2: sintering-beta
Input file name: Ti64-sintering-beta-350.input
Ti64-sintering-beta-350.input
Input file name: Ti64-sintering-beta-500.input
Ti64-sintering-beta-500.input
```

### 6.2.2 Solute C and O contents after heating in both layers

[25]:
```
# C_C_final_wtpc[node][debinding_condition]
C_C_final_wtpc=[]
C_O_final_wtpc=[]

for i in range(nnodes):
    C_C_final_wtpc.append([])
    C_O_final_wtpc.append([])
    for j in range(len(T_deb)):
```

```
        data=results[2][j].precipitation[i]

        XC=data["X_C"].iloc[-1]
        XO=data["X_O"].iloc[-1]
        C_C_final_wtpc[i].append(XC*M_C/(XC*M_C+XO*M_O+(1-XC-XO)*M_Ti)*100)
        C_O_final_wtpc[i].append(XO*M_O/(XC*M_C+XO*M_O+(1-XC-XO)*M_Ti)*100)
        #print(XC,XO)
#print(C_C_final_wtpc,C_O_final_wtpc)
#print(results[0][i].precipitation[1])
```

### 6.3 Cool down in $\beta$ to $T_{trans}$ (stage 3 )

[26]:
```
T_profile=[]
#T_trans=[]
T_trans=[600,600]
for i in range(len(T_deb)):
    #T_trans.append(slope_T_trans_O*C_O_final_wtpc[0][i]+intercept_T_trans_O)
    profile="0.01 "+str(T_sin[i]+TK)
    t_cool=0
    T_cool=T_sin[i]
    ti=tau_furnace*(math.log(T0/T_cool))**(1/n_furnace)
    T_cool=T_sin[i]-DeltaT_cool
    while(T_cool>T_trans[i]):
        deltat_cool=tau_furnace*(math.log(T0/T_cool))**(1/n_furnace) -ti
        profile = profile+" "+str(t_cool+deltat_cool)+" "+str(T_cool+TK)
        T_cool=T_cool-DeltaT_cool
    deltat_cool=tau_furnace*(math.log(T0/T_trans[i]))**(1/n_furnace) -ti
    profile = profile+" "+str(t_cool+deltat_cool)+" "+str(T_trans[i]+TK)
    T_profile.append(profile)
#print(T_profile)
```

#### 6.3.1 Fill the template file and run PreciSo

[27]:
```
results.append([])
condition="sintering-cool-beta"
nstage=3
stages.append(condition)
print("Stage "+str(nstage)+": "+condition)

for i in range(len(T_deb)):
    A=slope_A_TiC[1]*C_O_final_wtpc[0][i]+intercept_A_TiC[1]
    B=slope_B_TiC[1]*C_O_final_wtpc[0][i]+intercept_B_TiC[1]
    C=slope_C_TiC[1]*C_O_final_wtpc[0][i]+intercept_C_TiC[1]

    OptionalCommand="""
```

```
    initialDistrib TiC␣
↪distribution-Ti64-"""+stages[nstage-1]+"""-"""+str(T_deb[i])+"""TiC_0.dat
    classManagementType distrib"""

    # Key - Values to be inserted in the template
    values = {"systemName":systemName+"-"+condition,
          "nnodes":nnodes,
          "profile":T_profile[i],
          "C_Cwtpc_binder":C_C_final_wtpc[1][i],
          "C_Owtpc_binder":C_O_final_wtpc[1][i],
          "C_Cwtpc_bulk":C_C_final_wtpc[0][i],
          "C_Owtpc_bulk":C_O_final_wtpc[0][i],
          "C_Cwtpc_surf":C_C_final_wtpc[2][i],
          "C_Owtpc_surf":C_O_final_wtpc[2][i],
          "TinC":T_deb[i],
          "D0_C":D0_C_beta,
          "Q_C":Q_C_beta,
          "D0_O":D0_O_beta,
          "Q_O":Q_O_beta,
          "gamma":gamma_beta,
          "lattice_parameter":a_beta,
          "atomic_volume":v_at_beta,
          "A":A,
          "B":B,
          "C":C,
          "OptionalCommand":OptionalCommand}
    values.update(generic_values)

    # fill the template
    input_file = preciso.fillTemplate(input_template, values)

    # save the input file
    input_fileName=Path(systemName+'-'+condition+'-'+str(T_deb[i])+'.input')
    print("Input file name: "+str(input_fileName))
    with open(input_fileName, 'w') as f:
        f.write(input_file)
    print(input_fileName)
    # run PreciSo
    results[nstage].append(preciso.runSimulation(input_fileName, debug=␣
↪False,temp=False))
```

```
Stage 3: sintering-cool-beta
Input file name: Ti64-sintering-cool-beta-350.input
Ti64-sintering-cool-beta-350.input
Input file name: Ti64-sintering-cool-beta-500.input
Ti64-sintering-cool-beta-500.input
```

### 6.3.2 Solute C and O contents after heating in both layers

[28]:
```
# C_C_final_wtpc[node][debinding_condition]
C_C_final_wtpc=[]
C_O_final_wtpc=[]

for i in range(nnodes):
    C_C_final_wtpc.append([])
    C_O_final_wtpc.append([])
    for j in range(len(T_deb)):
        data=results[2][j].precipitation[i] # take results from stage 2 (no␣
 ↪precipitation)

        XC=data["X_C"].iloc[-1]
        XO=data["X_O"].iloc[-1]
        C_C_final_wtpc[i].append(XC*M_C/(XC*M_C+XO*M_O+(1-XC-XO)*M_Ti)*100)
        C_O_final_wtpc[i].append(XO*M_O/(XC*M_C+XO*M_O+(1-XC-XO)*M_Ti)*100)
        #print(XC,XO)
#print(C_C_final_wtpc,C_O_final_wtpc)
#print(results[0][i].precipitation[1])
```

## 6.4 Cooling in $\alpha$ from $T_{trans}$ to RT (stage 4 )

[29]:
```
T_profile=[]

for i in range(len(T_deb)):
    profile="0.01 "+str(T_trans[i]+TK)
    t_cool=0
    T_cool=T_trans[i]
    ti=tau_furnace*(math.log(T0/T_cool))**(1/n_furnace)
    T_cool=T_trans[i]-DeltaT_cool
    while(T_cool>RT):
        deltat_cool=tau_furnace*(math.log(T0/T_cool))**(1/n_furnace) -ti
        profile = profile+" "+str(t_cool+deltat_cool)+" "+str(T_cool+TK)
        T_cool=T_cool-DeltaT_cool
    deltat_cool=tau_furnace*(math.log(T0/RT))**(1/n_furnace) -ti
    profile = profile+" "+str(t_cool+deltat_cool)+" "+str(RT+TK)
    T_profile.append(profile)
#print(T_profile)
```

[30]:
```
condition="sintering-cool-alpha"
nstage=4
results.append([])
stages.append(condition)
print("Stage "+str(nstage)+": "+condition)

for i in range(len(T_deb)):
```

```
    A=slope_A_TiC[0]*C_O_final_wtpc[0][i]+intercept_A_TiC[0]
    B=slope_B_TiC[0]*C_O_final_wtpc[0][i]+intercept_B_TiC[0]
    C=slope_C_TiC[0]*C_O_final_wtpc[0][i]+intercept_C_TiC[0]

    OptionalCommand="""
    initialDistrib TiC␣
↪distribution-Ti64-"""+stages[nstage-1]+"""-"""+str(T_deb[i])+"""TiC_0.dat
    classManagementType distrib"""

    # Key - Values to be inserted in the template
    values = {"systemName":systemName+"-"+condition,
          "nnodes":nnodes,
          "profile":T_profile[i],
          "C_Cwtpc_binder":C_C_final_wtpc[1][i],
          "C_Owtpc_binder":C_O_final_wtpc[1][i],
          "C_Cwtpc_bulk":C_C_final_wtpc[0][i],
          "C_Owtpc_bulk":C_O_final_wtpc[0][i],
          "C_Cwtpc_surf":C_C_final_wtpc[2][i],
          "C_Owtpc_surf":C_O_final_wtpc[2][i],
          "TinC":T_deb[i],
          "D0_C":D0_C_alpha,
          "Q_C":Q_C_alpha,
          "D0_O":D0_O_alpha,
          "Q_O":Q_O_alpha,
          "gamma":gamma_alpha,
          "lattice_parameter":a_alpha,
          "atomic_volume":v_at_alpha,
          "A":A,
          "B":B,
          "C":C,
          "OptionalCommand":OptionalCommand}
    values.update(generic_values)

    # fill the template
    input_file = preciso.fillTemplate(input_template, values)

    # save the input file
    input_fileName=Path(systemName+'-'+condition+'-'+str(T_deb[i])+'.input')
    print("Input file name: "+str(input_fileName))
    with open(input_fileName, 'w') as f:
            f.write(input_file)
    print(input_fileName)
    # run PreciSo
    results[nstage].append(preciso.runSimulation(input_fileName, debug=␣
↪False,temp=False))
```

Stage 4: sintering-cool-alpha

```
Input file name: Ti64-sintering-cool-alpha-350.input
Ti64-sintering-cool-alpha-350.input
Input file name: Ti64-sintering-cool-alpha-500.input
Ti64-sintering-cool-alpha-500.input
```

# 7 Results

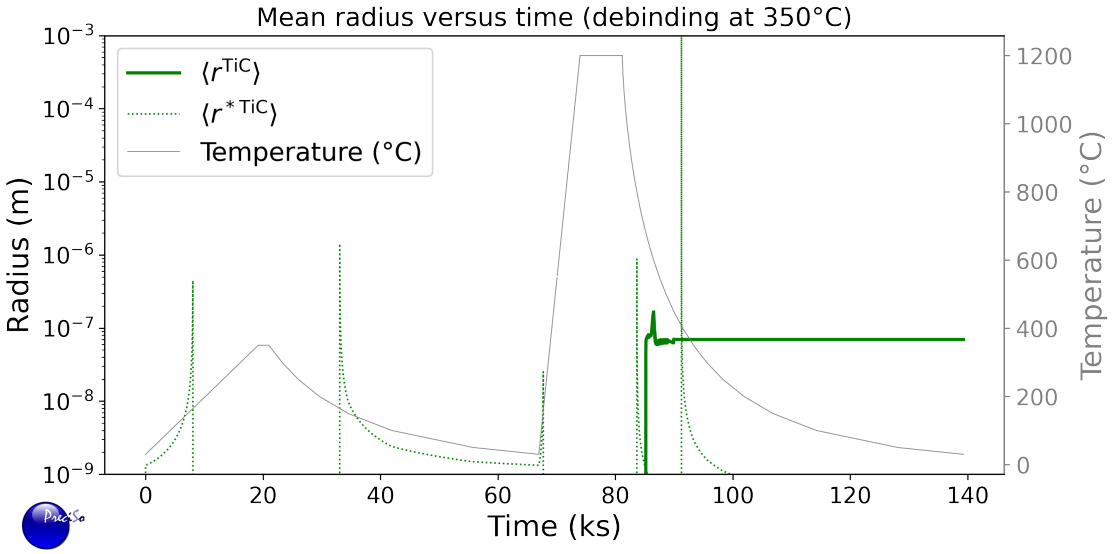
[31]:
```
# Display results on node 0
#results_debinding[0].precipitation[0]
print(T_deb)
```

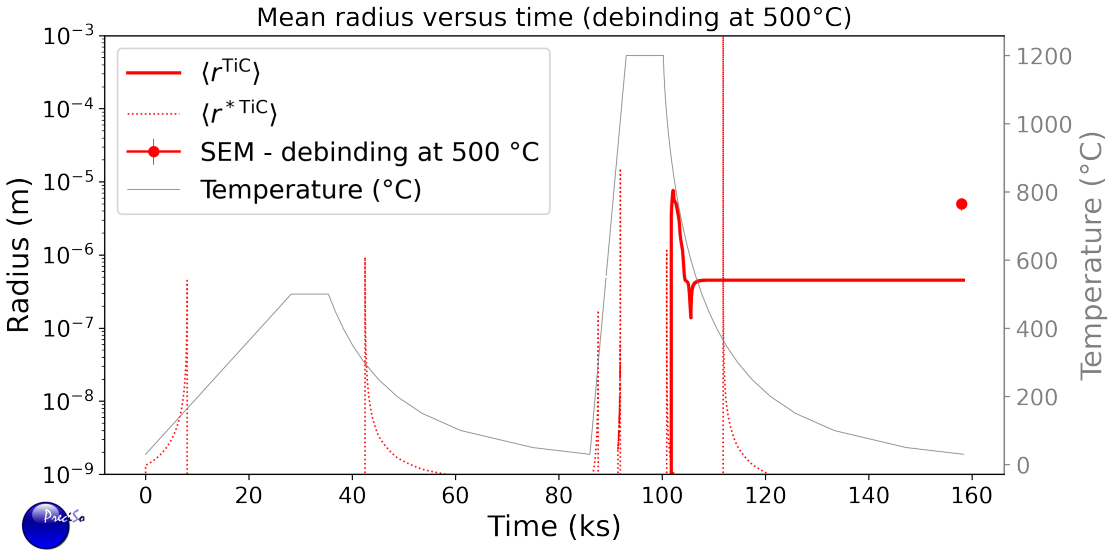
```
[350, 500]
```

## 7.1 Precipitate Radii

[32]:
```
T=T_deb
labelR=["$\\langle r^\\mathrm{TiC}\\rangle$","","","",""]
labelRstar=["$\\langle {r^*}^\\mathrm{TiC}\\rangle$","","","",""]
labelT=["Temperature (°C)","","","",""]
colorTemp='grey'


for i in range(len(T)):
    fig=plt.figure(figsize=(10,5))
    ax=fig.add_subplot(1,1,1)
    ax2 = ax.twinx()
    ax2.tick_params(labelsize=14)
    ax.tick_params(labelsize=14)
    maximum=0
    partial_time=0
    for n in range(len(stages)):
        data=results[n][i].precipitation[0]
        for j in range(len(precipitateName)):
            precipitateNameLabel=precipitateName[j]
            precipitateNameLabel=precipitateNameLabel.replace("gamma","\\gamma")
            ax.semilogy((data['t[s]']+partial_time)/1000,␣
 ↪data["rmean_"+precipitateName[j]+"[m]"],
                    color=colors[i],
                    label=labelR[n],
                    linewidth=2)
            ax.semilogy((data['t[s]']+partial_time)/1000,␣
 ↪data["r*_"+precipitateName[j]+"[m]"],
                    color=colors[i],
                    label=labelRstar[n],
                    linewidth=1, linestyle=':')
            ax2.plot((data['t[s]']+partial_time)/1000, data["T[K]"]-TK,
                    color=colorTemp, #ls='dashed', #marker="o",
```

```
                    label=labelT[n],
                    linewidth=0.5)
            partial_time+=data['t[s]'].iloc[-1]
            maximum=max(maximum,1.
↪5*max(data["rmean_"+precipitateName[j]+"[m]"]))

    dataPath=str(os.getcwd())+'/Radii-DS-'+str(T[i])+'/'
    ls = sorted(glob(dataPath + '*.txt'))
    #print(dataPath)
    #print(ls)
    markers = ["o", "^", ".", "x", "v", "<", ">", "d", "s","o", "^", ".", "x",␣
↪"v", "<", ">", "d", "s"]
    for k, file in enumerate(ls):
        label = os.path.splitext(os.path.split(file)[1])[0]
        dataToPlot=pd.read_csv(file,sep='\t',header =0)
        #print(dataToPlot)
        color=colors[i]
        ax.errorbar(dataToPlot["t[s]"]/1000, dataToPlot["r[m]"], color=color,␣
↪label=label, marker=markers[k],
                    yerr=1e-6,elinewidth = 0.5)
    plt.xticks(fontsize=16)
    ax.set_ylim([1e-9,1e-3])
    ax2.tick_params(axis='y', colors=colorTemp)
    ax2.spines['right'].set_color(colorTemp)
    ax2.yaxis.label.set_color(colorTemp)
    ax.set_ylabel('Radius (m)',fontsize=18)
    ax2.set_ylabel('Temperature (°C)',fontsize=18)
    ax.set_xlabel('Time (ks)',fontsize=18)
    plt.title('Mean radius versus time (debinding at␣
↪'+str(T[i])+'°C)',fontsize=16)
    fig.legend(loc='upper left', bbox_to_anchor=(0,1), bbox_transform=ax.
↪transAxes,fontsize=16)
    preciso.add_logo(fig,x_frac=0.05, y_frac=0.05, scale=0.25, alpha=1)
    plt.savefig('RadiiVsTime'+str(T[i])+'.pdf',bbox_inches='tight')
```



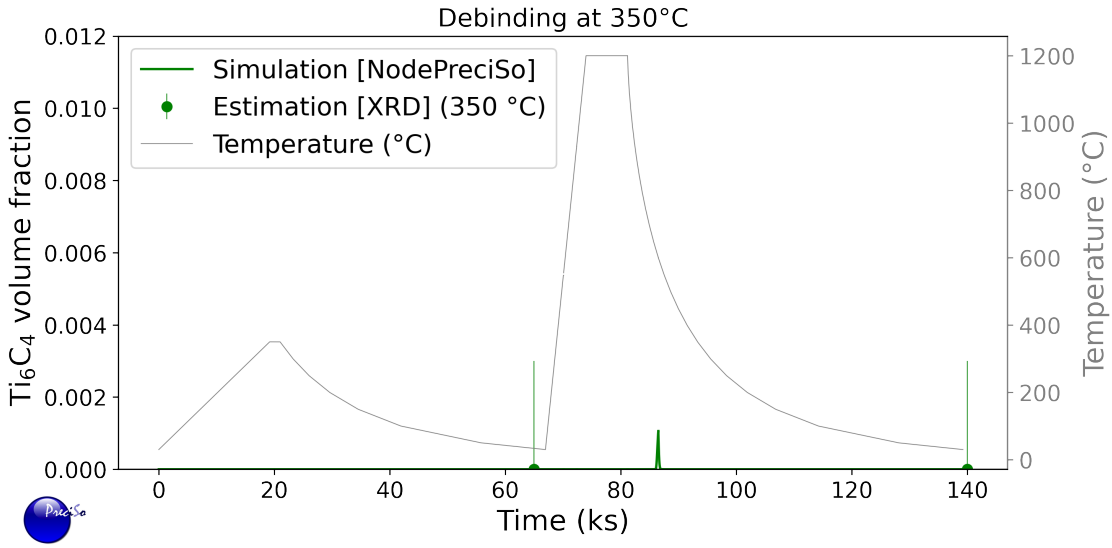

### 7.1.1 Precipitate volume fraction

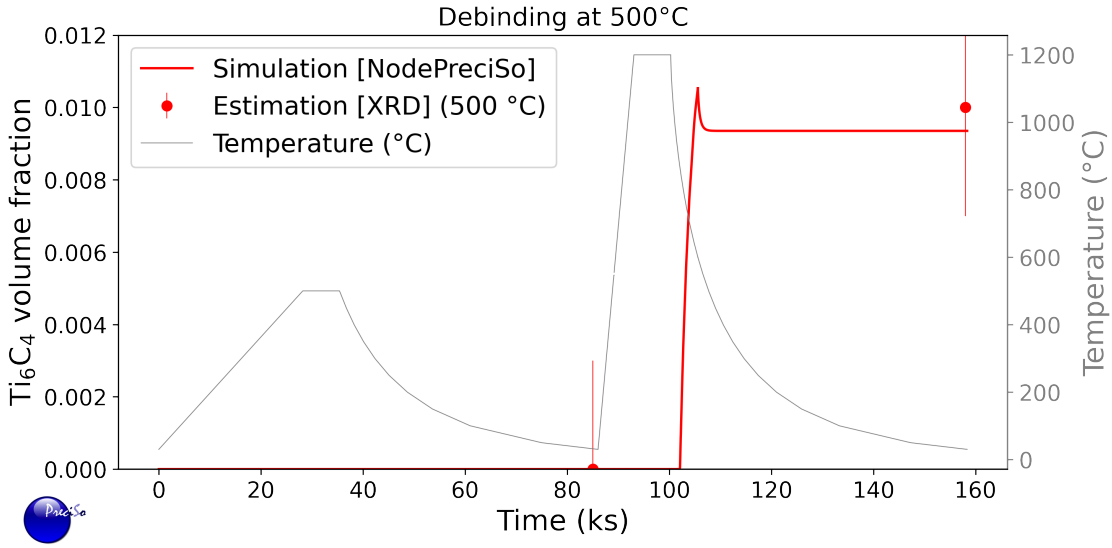
[42]:
```
T=T_deb
labelfv=["Simulation [NodePreciSo]","","","","",""]
labelT=["Temperature (°C)","","","","",""]

for i in range(len(T)):
    fig=plt.figure(figsize=(10,5))
    ax=fig.add_subplot(1,1,1)
    ax2 = ax.twinx()
```

```
    ax2.tick_params(labelsize=14)
    ax.tick_params(labelsize=14)
    maximum=0
    partial_time=0
    for n in range(len(stages)):
        data=results[n][i].precipitation[0]
        for j in range(len(precipitateName)):
            precipitateNameLabel=precipitateName[j]
            precipitateNameLabel=precipitateNameLabel.replace("gamma","\\gamma")
            ax.plot((data['t[s]']+partial_time)/1000,␣
↪data["fv_"+precipitateName[j]], color=colors[i],
                 label=labelfv[n])
            ax2.plot((data['t[s]']+partial_time)/1000, data["T[K]"]-TK,
                   color=colorTemp, #ls='dashed', #marker="o",
                   label=labelT[n],
                   linewidth=0.5)
            maximum=max(maximum,1.1*max(data["fv_"+precipitateName[j]]))
        partial_time+=data['t[s]'].iloc[-1]

    dataPath=str(os.getcwd())+'/Fv-'+str(T[i])+'/'
    ls = sorted(glob(dataPath + '*.txt'))
    markers = ["o", "^", ".", "x", "v", "<", ">", "d", "s"]
    for k, file in enumerate(ls):
        label = os.path.splitext(os.path.split(file)[1])[0]
        dataToPlot=pd.read_csv(file,sep='\t',header =0)
        print(dataToPlot)
        if len(ls) == 1:
            color = 'blue'
        else:
            color=cm.jet(i*1./(len(ls)-1))
        ax.errorbar(dataToPlot["t[s]"]/1000, dataToPlot["fv"], color=colors[i],␣
↪linestyle='None',
                    yerr=0.003,label=label, marker=markers[k],elinewidth = 0.5)

    #ax.arrow(7, 0.002, 10, 0, color =colors[i],width = 0.00001,ls=␣
↪'dashed',head_width=0.0002, head_length=2)
    ax2.tick_params(axis='y', colors=colorTemp)
    ax2.spines['right'].set_color(colorTemp)
    ax2.yaxis.label.set_color(colorTemp)
    ax.set_ylabel('Ti$_6$C$_4$ volume fraction',fontsize=18)
    ax2.set_ylabel('Temperature (°C)',fontsize=18)
    ax.set_ylim([0,0.012])
    #ax.set_xlim([50879,50882])
    ax.set_xlabel('Time (ks)',fontsize=18)
    plt.title('Debinding at '+str(T[i])+'°C',fontsize=16)
    fig.legend(loc='upper left', bbox_to_anchor=(0,1), bbox_transform=ax.
↪transAxes,fontsize=16)
```

```
    preciso.add_logo(fig,x_frac=0.05, y_frac=0.05, scale=0.25, alpha=1)
    plt.savefig('VolFractionVsTime'+str(T[i])+'.pdf',bbox_inches='tight')
```

```
     t[s]  fv
0   65000   0
1  140000   0
     t[s]    fv
0   85000  0.00
1  158000  0.01
```

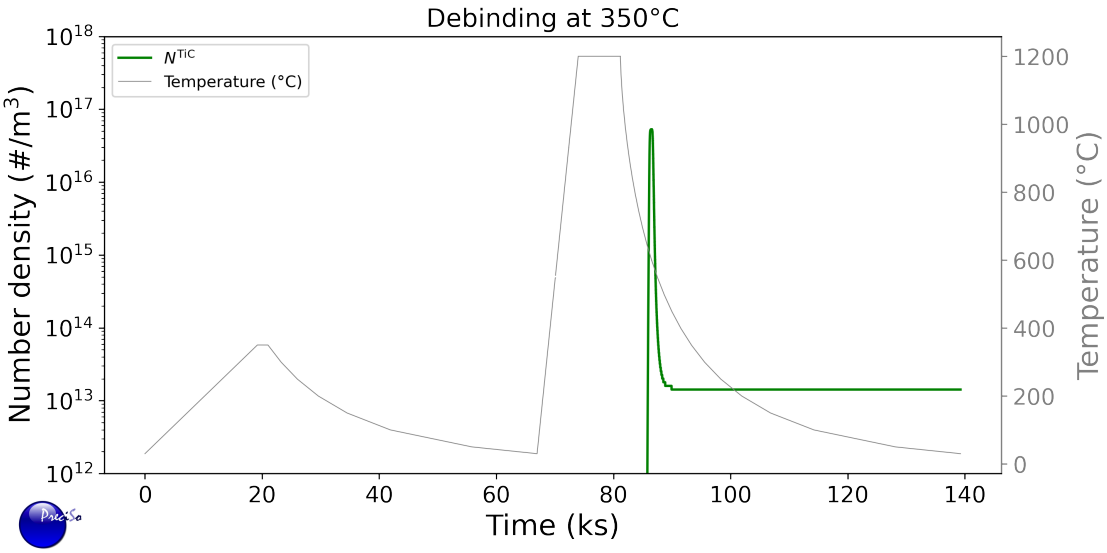


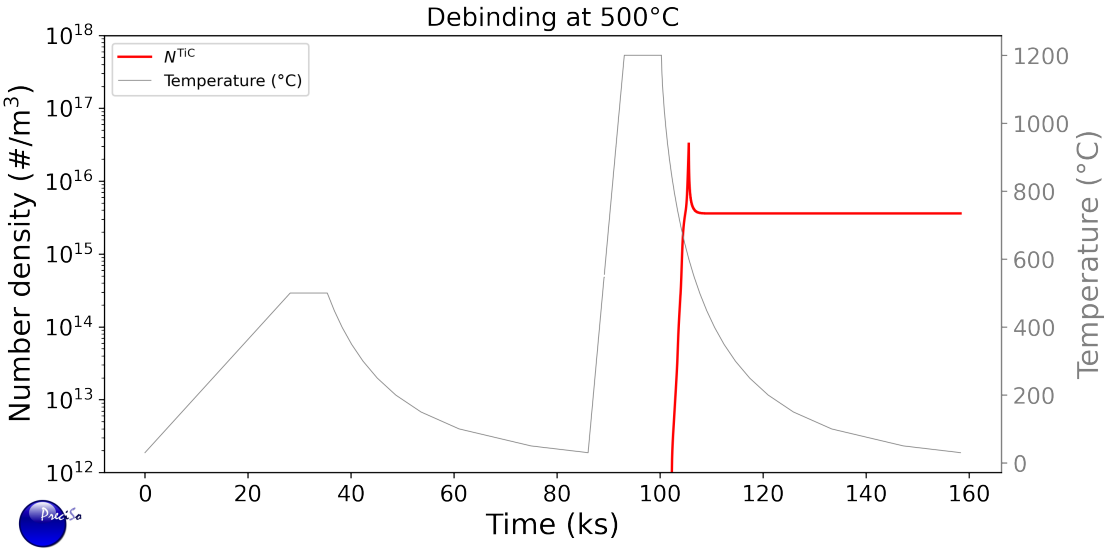



### 7.1.2 Precipitate number density

[34]:
```
T=T_deb
labelN=["$N^\mathrm{TiC}$","","","",""]
labelT=["Temperature (°C)","","","",""]

for i in range(len(T)):
    fig=plt.figure(figsize=(10,5))
    ax=fig.add_subplot(1,1,1)
    ax2 = ax.twinx()
    ax2.tick_params(labelsize=14)
    ax.tick_params(labelsize=14)
    maximum=0
    partial_time=0
    for n in range(len(stages)):
        data=results[n][i].precipitation[0]
        for j in range(len(precipitateName)):
            precipitateNameLabel=precipitateName[j]
            precipitateNameLabel=precipitateNameLabel.replace("gamma","\\gamma")
            ax.semilogy((data['t[s]']+partial_time)/1000,␣
 ↪data["N_"+precipitateName[j]], color=colors[i],
                 label=labelN[n])
            ax2.plot((data['t[s]']+partial_time)/1000, data["T[K]"]-TK,
                   color=colorTemp, #ls='dashed', #marker="o",
                   label=labelT[n],
                   linewidth=0.5)
            maximum=max(maximum,1.1*max(data["N_"+precipitateName[j]]))
        partial_time+=data['t[s]'].iloc[-1]

    dataPath=str(os.getcwd())+'/ND-DS-'+str(T[i])+'/'
    ls = sorted(glob(dataPath + '*.txt'))
    markers = ["o", "^", ".", "x", "v", "<", ">", "d", "s"]
    for k, file in enumerate(ls):
        label = os.path.splitext(os.path.split(file)[1])[0]
        dataToPlot=pd.read_csv(file,sep='\t',header =0)
        print(dataToPlot)
        if len(ls) == 1:
            color = 'blue'
        else:
            color=cm.jet(i*1./(len(ls)-1))
        plt.scatter((dataToPlot["t[s]"])/1000, dataToPlot["fv"],␣
 ↪color=colors[i], label=label, marker=markers[k], s=50)

    ax.set_ylim([1e12,1e18])
    ax2.tick_params(axis='y', colors=colorTemp)
```

```
    ax2.spines['right'].set_color(colorTemp)
    ax2.yaxis.label.set_color(colorTemp)
    ax.set_ylabel('Number density (#/m$^3$)',fontsize=18)
    ax2.set_ylabel('Temperature (°C)',fontsize=18)
    ax.set_xlabel('Time (ks)',fontsize=18)
    plt.title('Debinding at '+str(T[i])+'°C',fontsize=16)
    fig.legend(loc='upper left', bbox_to_anchor=(0,1), bbox_transform=ax.
↪transAxes)
    preciso.add_logo(fig,x_frac=0.05, y_frac=0.05, scale=0.25, alpha=1)
    plt.savefig('NumberDensityVsTime'+str(T[i])+'.pdf',bbox_inches='tight')
```



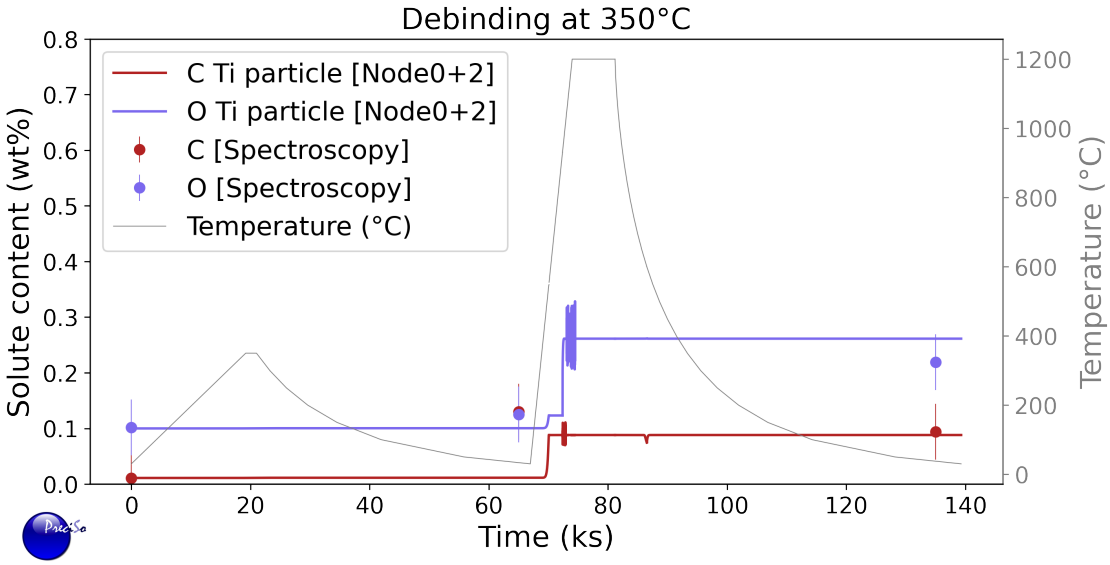
### 7.1.3 C and O solute concentrations

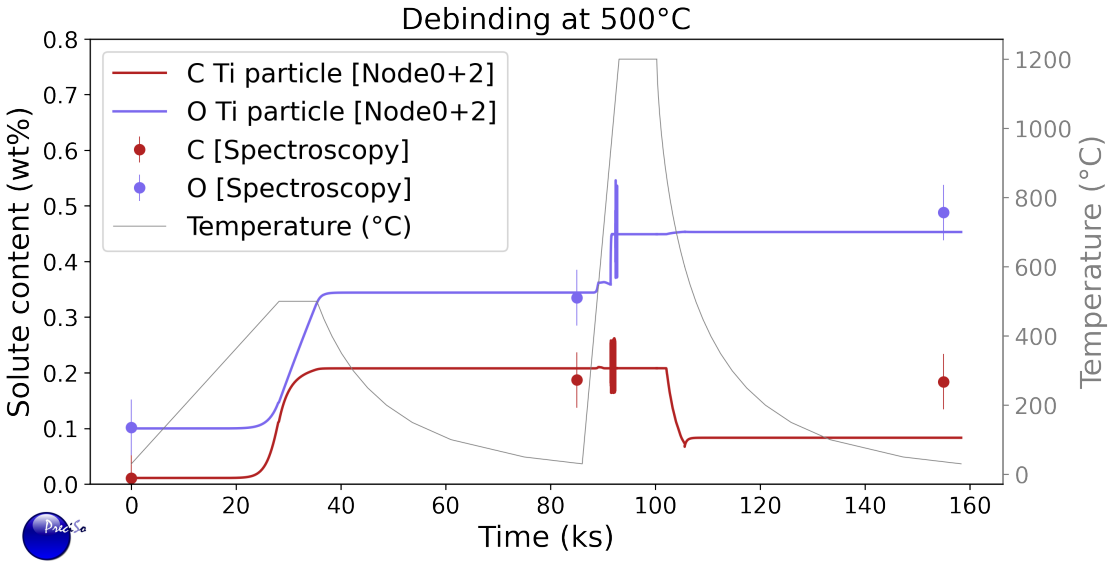
[35]:
```
T=T_deb
labelXC=["C Ti particle [Node0+2]","","","",""]
labelXC0=["C bulk [Node0]","","","",""]
labelXC2=["C surface [Node2]","","","",""]
labelXO=["O Ti particle [Node0+2]","","","",""]
labelXO0=["O bulk [Node0]","","","",""]
labelXO2=["O surface [Node2]","","","",""]
labelT=["Temperature (°C)","","","",""]

for i in range(len(T)):
    fig=plt.figure(figsize=(10,5))
    ax=fig.add_subplot(1,1,1)
    ax2 = ax.twinx()
    ax2.tick_params(labelsize=14)
    ax.tick_params(labelsize=14)
    maximum=0
    partial_time=0
    for n in range(len(stages)):
        data0=results[n][i].precipitation[0]
        data1=results[n][i].precipitation[1]
        data2=results[n][i].precipitation[2]
        Cwtpc0=data0["X_C"]*M_C/
↪(data0["X_C"]*M_C+data0["X_O"]*M_O+(1-data0["X_C"]-data0["X_O"])*M_Ti)*100
        Owtpc0=data0["X_O"]*M_O/
↪(data0["X_C"]*M_C+data0["X_O"]*M_O+(1-data0["X_C"]-data0["X_O"])*M_Ti)*100
        Cwtpc1=data1["X_C"]*M_C/
↪(data1["X_C"]*M_C+data1["X_O"]*M_O+(1-data1["X_C"]-data1["X_O"])*M_Ti)*100
        Owtpc1=data1["X_O"]*M_O/
↪(data1["X_C"]*M_C+data1["X_O"]*M_O+(1-data1["X_C"]-data1["X_O"])*M_Ti)*100
        Cwtpc2=data2["X_C"]*M_C/
↪(data2["X_C"]*M_C+data2["X_O"]*M_O+(1-data2["X_C"]-data2["X_O"])*M_Ti)*100
        Owtpc2=data2["X_O"]*M_O/
↪(data2["X_C"]*M_C+data2["X_O"]*M_O+(1-data2["X_C"]-data2["X_O"])*M_Ti)*100
        Cwtpc=(Cwtpc0*Vol_Node0+Cwtpc2*Vol_Node2)/(Vol_Node0+Vol_Node2)
        Owtpc=(Owtpc0*Vol_Node0+Owtpc2*Vol_Node2)/(Vol_Node0+Vol_Node2)
        #ax.plot((data0['t[s]']+partial_time)/1000, Cwtpc0, color="blue",␣
↪label=labelXC0[n])
        #ax.plot((data2['t[s]']+partial_time)/1000, Cwtpc2, color="blue",␣
↪label=labelXC2[n])
        ax.plot((data0['t[s]']+partial_time)/1000, Cwtpc, color=colorsChem[0],␣
↪label=labelXC[n])
```

```
        ax.plot((data0['t[s]']+partial_time)/1000, Owtpc, color=colorsChem[1],␣
↪label=labelXO[n])
        ax2.plot((data0['t[s]']+partial_time)/1000, (data0["T[K]"]-TK),
                    color=colorTemp, #ls='dashed',# marker="o",
                    label=labelT[n],
                    linewidth=0.5)
        maximum=max(maximum,1.1*max(data0["X_C"]))
        partial_time+=data0['t[s]'].iloc[-1]

    dataPath=str(os.getcwd())+'/SC-'+str(T[i])+'/'
    ls = sorted(glob(dataPath + '*.txt'))
    markers = ["o", "^", ".", "x", "v", "<", ">", "d", "s"]
    for k, file in enumerate(ls):
        label = os.path.splitext(os.path.split(file)[1])[0]
        dataToPlot=pd.read_csv(file,sep='\t',header =0)
        print(dataToPlot)
        #if(label=='C [Spectroscopy]'):
        ax.errorbar(dataToPlot["t[s]"]/1000, dataToPlot["SC"],␣
↪color=colorsChem[k], linestyle='None',
                    yerr=0.05,label=label, marker="o",elinewidth = 0.5)
    plt.xticks(fontsize=16)
    ax.set_ylim([0,0.8])
    #ax.set_xlim([50879,50882])
    #ax.set_ylabel('Solute content (at. frac.)')
    ax2.tick_params(axis='y', colors=colorTemp)
    ax2.spines['right'].set_color(colorTemp)
    ax2.yaxis.label.set_color(colorTemp)
    ax.set_ylabel('Solute content (wt%)',fontsize=18)
    ax2.set_ylabel('Temperature (°C)',fontsize=18)
    ax.set_xlabel('Time (ks)',fontsize=18)
    plt.title('Debinding at '+str(T[i])+'°C',fontsize=18)

    fig.legend(loc='upper left', bbox_to_anchor=(0,1), bbox_transform=ax.
↪transAxes,fontsize=16)
    #ax2.legend(loc='best')
    preciso.add_logo(fig,x_frac=0.05, y_frac=0.05, scale=0.25, alpha=1)
    plt.savefig('SoluteContentVsTime'+str(T[i])+'.pdf',bbox_inches='tight')
    #plt.savefig('SoluteContentVsTime-noPrecipitation'+str(T[i])+'.
↪pdf',bbox_inches='tight')
```

```
     t[s]     SC
0       0  0.011
1   65000  0.130
2  135000  0.094
     t[s]     SC
0       0  0.102
```

```
1    65000   0.125
2   135000   0.219
      t[s]      SC
0        0   0.011
1    85000   0.187
2   155000   0.184
      t[s]      SC
0        0   0.102
1    85000   0.335
2   155000   0.488
```



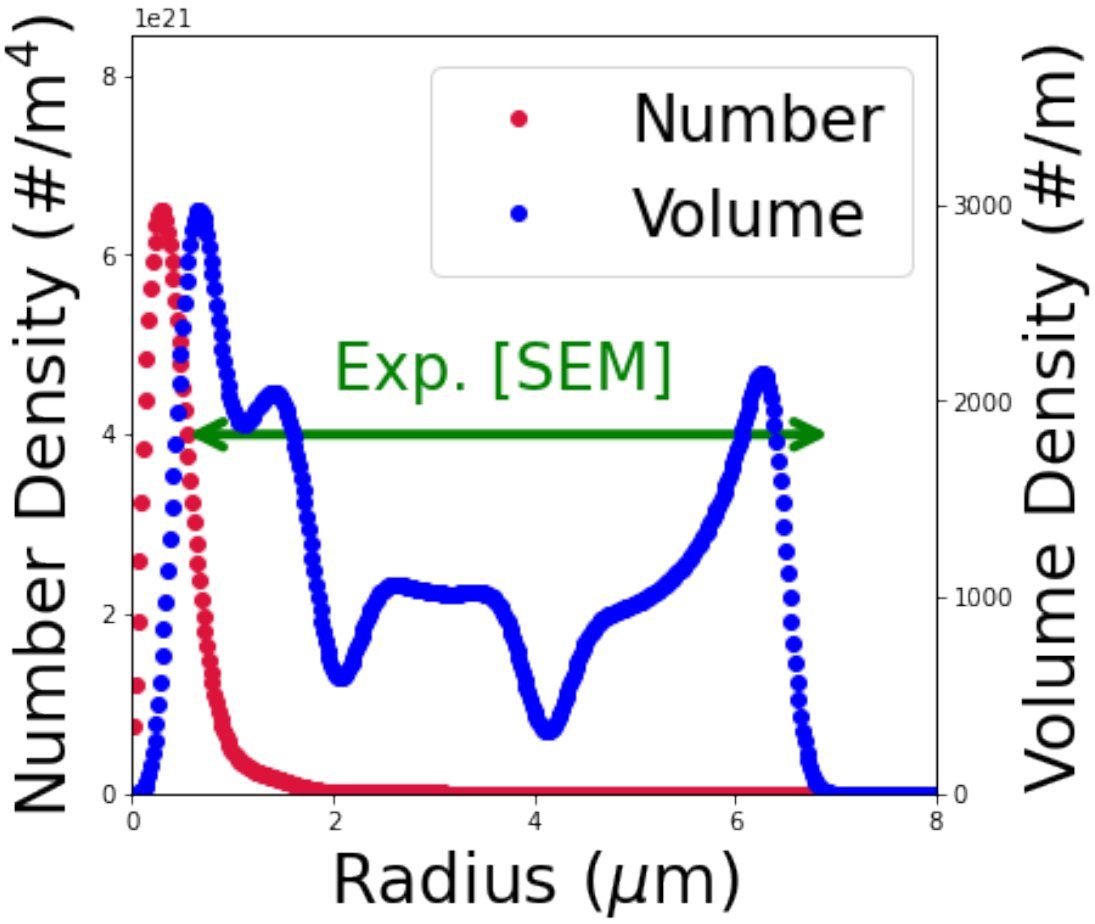



## 7.2 Distribution

[36]:
```
fig=plt.figure(figsize=(7,6))
ax2.tick_params(labelsize=26)
ax.tick_params(labelsize=26)
nstage=4

ax=fig.add_subplot(1,1,1)
ax2 = ax.twinx()

last_step=list(results[nstage][1].distribution.data[0]["TiC"])[-1]
print(last_step)
data=results[nstage][1].distribution.data[0]["TiC"][last_step]

ax2.plot(data['R_i']*1e6,data["D_i"]*4/3*math.
 ↪pi*data['R_i']**3,'o',color='blue', label="Volume")
ax.plot(data['R_i']*1e6,data["D_i"],'o',color='crimson', label="Number")
R_exp_min=1e-6
R_exp_max=10e-6


y_max=max(data["D_i"])
y2_max=max(4/3*math.pi*data["D_i"]*data['R_i']**3)


#plt.bar(R_exp_min, y_max, color="green", linewidth=0,␣
 ↪align="center",width=1e-7)
plt.rc('font', size=26)

ax.annotate('', xy=(7, 4e21), xytext=(0.5, 4e21),
            arrowprops=dict(color='green',arrowstyle='<->',linewidth=4)
            )
ax.text(2,4.5e21,r'Exp. [SEM]',{'color': 'green'},fontsize=26)
#plt.text(mean_radius*1.2,max(data["D_i"])*0.5,r'$\langle R␣
 ↪\rangle_\mathrm{TEM}$',{'color': 'blue'})

plt.xlim([0,8])
ax.set_ylim([0,y_max*1.3])
ax2.set_ylim([0,y2_max*1.3])

ax.set_ylabel('Number Density (#/m$^4$)',fontsize=28)
ax2.set_ylabel('Volume Density (#/m)',fontsize=28)
ax.set_xlabel('Radius ($\mu$m)',fontsize=28)
#plt.title("Distribution",fontsize=26)
fig.legend(loc='upper right', bbox_to_anchor=(1.01,1), bbox_transform=ax.
 ↪transAxes,fontsize=26)
```

```
plt.gcf().tight_layout()
#preciso.add_logo(fig,x_frac=0.05, y_frac=0.05, scale=0.3, alpha=1,)
plt.savefig('Distributions.pdf',bbox_inches='tight')
```

87




### 7.2.1 Zener/Rios Pinning pressure and limiting grain diameter

[37]:
```
nstage=4
last_step=list(results[nstage][1].distribution.data[0]["TiC"])[-1]
data=results[nstage][1].distribution.data[0]["TiC"][last_step]
alpha_Rios=3/2
alpha_Zener=3
beta=4
R=data['R_i']
```

```
N=data['N_i']
S_NR2=0
#P_d=beta/D
for i in range(len(R)):
    S_NR2+=N[i]*R[i]*R[i]
#P_p=S_NR2*4/3*math.pi*alpha

D_lim_Zener=3*beta/4/math.pi/alpha_Zener/S_NR2
D_lim_Rios=3*beta/4/math.pi/alpha_Rios/S_NR2

print("D_lim_Zener ($\mu$m): ", D_lim_Zener*1e6)
print("D_lim_Rios ($\mu$m): ", D_lim_Rios*1e6)
```

```
D_lim_Zener ($\mu$m):  251.1737043129834
D_lim_Rios ($\mu$m):  502.3474086259668
```

# 8 Isothermal precipition in $\alpha$ phase (stage #5)

## 8.1 Parameters

[38]:
```
C_O_wtpc=0.5
T_iso=500
t_iso=1e8
C_C_wtpc=1
gamma=[0.2,0.21,0.22,0.23,0.24,0.25]
```

### 8.1.1 Fill the template file and run PreciSo

[39]:
```
condition="isothermal-alpha"
nstage=5
if(len(results)<nstage+1):
    results.append([])
    stages.append(condition)
print("Stage "+str(nstage)+": "+condition)

OptionalCommand="""
classManagementType old"""

for i in range(len(gamma)):
    A=slope_A_TiC[0]*C_O_wtpc+intercept_A_TiC[0]
    B=slope_B_TiC[0]*C_O_wtpc+intercept_B_TiC[0]
    C=slope_C_TiC[0]*C_O_wtpc+intercept_C_TiC[0]


    # Key - Values to be inserted in the template
    values = {"systemName":systemName+"-"+condition,
          "nnodes":nnodes,
```

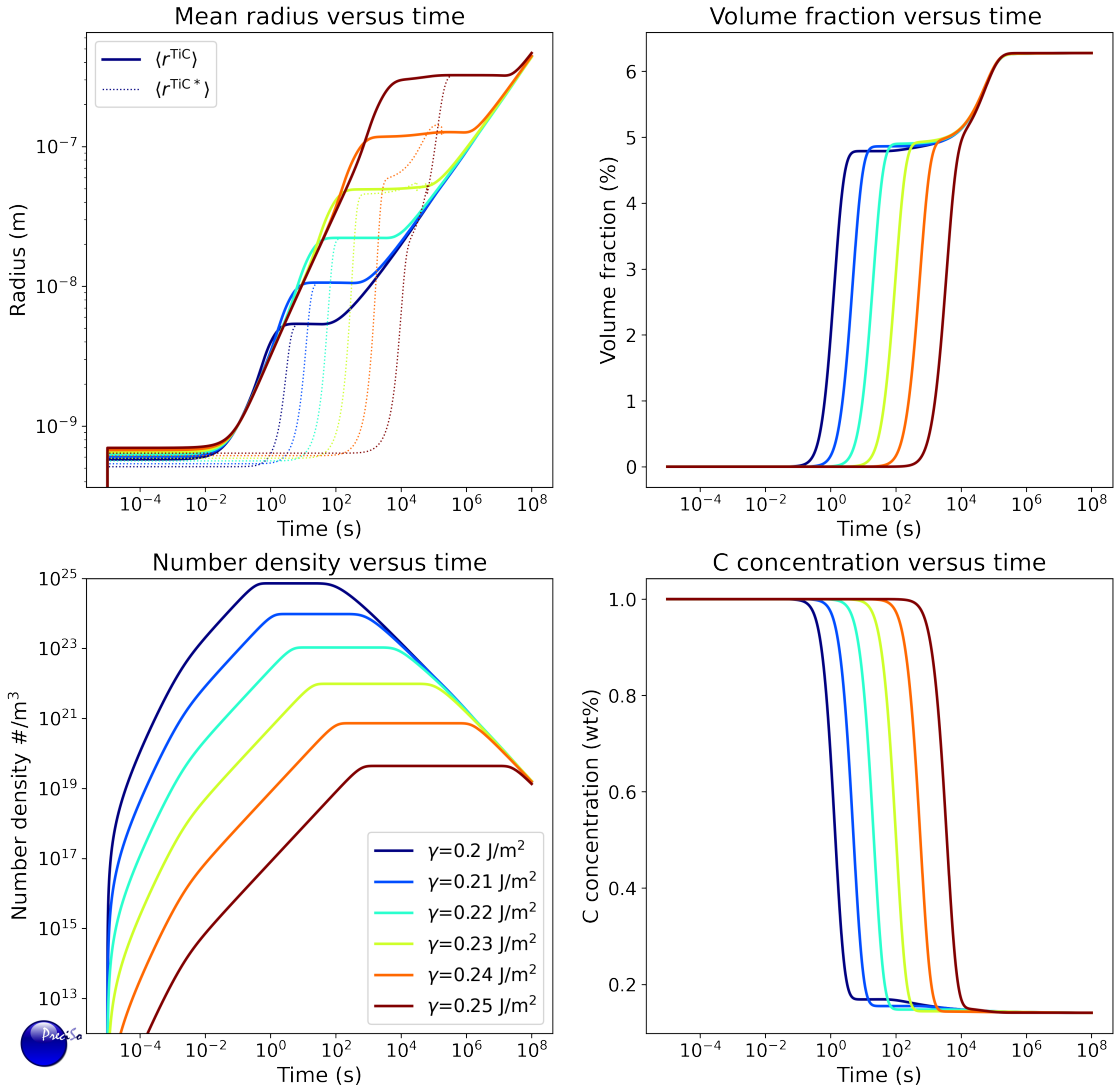
```
        "profile":"0.00001 "+str(T_iso+TK)+" "+str(t_iso)+" "+str(T_iso+TK),
        "C_Cwtpc_binder":0,
        "C_Owtpc_binder":0,
        "C_Cwtpc_bulk":C_C_wtpc,
        "C_Owtpc_bulk":C_O_wtpc,
        "C_Cwtpc_surf":C_C_wtpc,
        "C_Owtpc_surf":C_O_wtpc,
        "TinC":T_iso,
        "D0_C":D0_C_alpha,
        "Q_C":Q_C_alpha,
        "D0_O":D0_O_alpha,
        "Q_O":Q_O_alpha,
        "gamma":gamma[i],
        "lattice_parameter":a_alpha,
        "atomic_volume":v_at_alpha,
        "A":A,
        "B":B,
        "C":C,
        "OptionalCommand":OptionalCommand}
    values.update(generic_values)

    # fill the template
    input_file = preciso.fillTemplate(input_template, values)

    # save the input file
  ␣
↪input_fileName=Path(systemName+'-'+condition+'-'+str(T_iso)+'-gamma_'+str(gamma[i])+'.
↪input')
    print("Input file name: "+str(input_fileName))
    with open(input_fileName, 'w') as f:
            f.write(input_file)
    print(input_fileName)
    # run PreciSo
    if(len(results[nstage])<len(gamma)):
        results[nstage].append(preciso.runSimulation(input_fileName, debug=␣
↪False,temp=False))
```

```
Stage 5: isothermal-alpha
Input file name: Ti64-isothermal-alpha-500-gamma_0.2.input
Ti64-isothermal-alpha-500-gamma_0.2.input
Input file name: Ti64-isothermal-alpha-500-gamma_0.21.input
Ti64-isothermal-alpha-500-gamma_0.21.input
Input file name: Ti64-isothermal-alpha-500-gamma_0.22.input
Ti64-isothermal-alpha-500-gamma_0.22.input
Input file name: Ti64-isothermal-alpha-500-gamma_0.23.input
Ti64-isothermal-alpha-500-gamma_0.23.input
Input file name: Ti64-isothermal-alpha-500-gamma_0.24.input
```

```
Ti64-isothermal-alpha-500-gamma_0.24.input
Input file name: Ti64-isothermal-alpha-500-gamma_0.25.input
Ti64-isothermal-alpha-500-gamma_0.25.input
```

[40]:
```
T=T_iso
n = len(gamma)
colors_jet = pl.cm.jet(np.linspace(0,1,n))
fig,((ax11,ax12),(ax21,ax22))=plt.subplots(2,2,figsize=(14,14))
ax11.tick_params(labelsize=14)
ax12.tick_params(labelsize=14)
ax21.tick_params(labelsize=14)
ax22.tick_params(labelsize=14)


n=5 #stage 5



for i in range(len(gamma)):
    maximum=0
    data=results[n][i].precipitation[0]

    # Radius
    if i==0:
        label_r="$\\langle {r^\mathrm{TiC}}\\rangle$"
        label_r_star="$\\langle {r^\mathrm{TiC}}^*\\rangle$"
    else:
        label_r=""
        label_r_star=""
    for j in range(len(precipitateName)):
            precipitateNameLabel=precipitateName[j]
            precipitateNameLabel=precipitateNameLabel.replace("gamma","\\gamma")
            ax11.loglog(data['t[s]'], data["rmean_"+precipitateName[j]+"[m]"],␣
↪color=colors_jet[i],
                    label=label_r, linewidth=2)
            ax11.loglog(data['t[s]'], data["r*_"+precipitateName[j]+"[m]"],␣
↪color=colors_jet[i],
                    label=label_r_star, linewidth=1, linestyle=':')
            maximum=max(maximum,1.
↪5*max(data["rmean_"+precipitateName[j]+"[m]"]))
    ax11.set_ylabel('Radius (m)',fontsize=16)
    ax11.set_xlabel('Time (s)',fontsize=16)
    ax11.set_title('Mean radius versus time',fontsize=18)
    ax11.legend(fontsize=14)

    # Volume Fraction
    maximum=0
```

```
    data=results[n][i].precipitation[0]
    for j in range(len(precipitateName)):
            precipitateNameLabel=precipitateName[j]
            precipitateNameLabel=precipitateNameLabel.replace("gamma","\\gamma")
            ax12.semilogx(data['t[s]'], data["fv_"+precipitateName[j]]*100,
                   color=colors_jet[i],
                   label="$\gamma$="+str(gamma[i])+" J/m$^2$",
                   linewidth=2)
            maximum=max(maximum,1.5*max(data["fv_"+precipitateName[j]]))
    ax12.set_ylabel('Volume fraction (%)',fontsize=16)
    ax12.set_xlabel('Time (s)',fontsize=16)
    ax12.set_title('Volume fraction versus time ',fontsize=18)

    # Number density
    for j in range(len(precipitateName)):
            precipitateNameLabel=precipitateName[j]
            precipitateNameLabel=precipitateNameLabel.replace("gamma","\\gamma")
            ax21.loglog(data['t[s]'], data["N_"+precipitateName[j]]*100,
                   color=colors_jet[i],
                   label="$\gamma$="+str(gamma[i])+" J/m$^2$",
                   linewidth=2)
            maximum=max(maximum,1.5*max(data["N_"+precipitateName[j]]))
    ax21.set_ylim([1e12,1e25])
    ax21.set_ylabel('Number density #/m$^3$',fontsize=16)
    ax21.set_xlabel('Time (s)',fontsize=16)
    ax21.set_title('Number density versus time',fontsize=18)
    ax21.legend(fontsize=14)

    # Solute concentration
    for j in range(len(precipitateName)):
            precipitateNameLabel=precipitateName[j]
            precipitateNameLabel=precipitateNameLabel.replace("gamma","\\gamma")
            Cwtpc=data["X_C"]*M_C/
↪(data["X_C"]*M_C+data["X_O"]*M_O+(1-data["X_C"]-data["X_O"])*M_Ti)*100
            ax22.semilogx(data['t[s]'], Cwtpc,
                   color=colors_jet[i],
                   label="$\gamma$="+str(gamma[i])+" J/m$^2$",
                   linewidth=2)
            maximum=max(maximum,1.5*max(Cwtpc))
    ax22.set_ylabel('C concentration (wt%)',fontsize=16)
    ax22.set_xlabel('Time (s)',fontsize=16)
    ax22.set_title('C concentration versus time',fontsize=18)


#plt.title('Isothermal treatment at '+str(T_iso)+'°C)')
preciso.add_logo(fig,x_frac=0.04, y_frac=0.04, scale=0.3, alpha=1)
plt.savefig('Isothermal-'+str(T_iso)+'.pdf',bbox_inches='tight')
```


Mean radius versus time
$\langle r^{TiC} \rangle$
$\langle r^{TiC*} \rangle$
Radius (m)
Time (s)
Volume fraction versus time
Volume fraction (%)
Time (s)
Number density versus time
Number density #/m$^3$
$\gamma$=0.2 J/m$^2$
$\gamma$=0.21 J/m$^2$
$\gamma$=0.22 J/m$^2$
$\gamma$=0.23 J/m$^2$
$\gamma$=0.24 J/m$^2$
$\gamma$=0.25 J/m$^2$
Time (s)
C concentration versus time
C concentration (wt%)
Time (s)

[ ]: