An efficient and versatile use of the VCCT for composites delamination growth under fatigue loadings in 3D numerical analysis: the Sequential Static Fatigue algorithm

L. M. Martulli[1]*, A. Bernasconi [1]

[1] Politecnico di Milano, Mechanical Engineering Department, Via La Masa 1, 20156 Milano, Italy

*Corresponding author: lucamichele.martulli@polimi.it

Abstract

Composite materials are susceptible to delamination when undergoing fatigue loads. Being able to accurately and efficiently predict the fatigue propagation of a delamination defect would increase the safety and the adoption of lightweight composites within industry. In this work, a new VCCT-based 3D fatigue propagation algorithm, called Sequential-Static Fatigue (SSF), is presented. The algorithm is able to accurately simulate the fatigue propagation of delamination defects by performing several sequential static simulations. Compared to the benchmark (Abaqus direct cyclic algorithm), the SSF reduced the computational time of several cases by three orders of magnitude. A better accuracy was also achieved. This work thus sets the basis for a new drastically more efficient modelling technique for fatigue delamination propagation.

Keywords: Fatigue; Delamination; Composite materials; Finite element analysis; Fracture mechanics.

1 Introduction

One of the most dangerous and common forms of damage in laminated composites and adhesively bonded joints is delamination propagation under fatigue loads [1]. Delaminations are cracks nucleating and propagating between the layers of composite laminates. The widespread use of composites and/or structural bonding in several industries has made the ability to simulate the evolution of such defects of critical importance. The most commonly adopted available options for this purpose are the Cohesive Zone Models (CZMs) and the Virtual Crack Closure Technique (VCCT) [2].

Research works based on CZMs are more abundant [3–8]. CZMs are based on the discretisation of the interface between two adjacent layers via interface elements. These methods were shown to lead to accurate results with reasonable computational times [4,5]. Compared to VCCT, they do not require an

initial crack presence and are generally less mesh sensitive [2]. However, CZMs usually require a traction-separation law that describes the bonding in terms of fracture toughness, initial stiffness and damage evolution. Different cohesive laws have been proposed in the literature, and the choice is often a trade-off between accuracy and computational efficiency. Moreover, these laws make use of several fitting parameters, while other physically based ones are often difficult to obtain via experimental tests. The development of fatigue crack propagation models based on CZMs is thus even more complex, as a degradation law needs to be developed and implemented.

VCCT is a fracture mechanics-based approach [9,10]. It was used with good results to predict quasi-static maximum loads, strain energy release rate distribution at crack tips and has been the object of several research works in general [11–16]. Combining VCCT with Paris-like fatigue crack growth models, the technique can be used for fatigue crack propagation [17]. VCCT offers some advantages over CZMs, above all its simpler implementation [5,9,10,18]. This is likely the reason why commercially available Finite Element (FE) software have had a dedicated tool for fatigue crack propagation using VCCT since more than a decade [19,20], while commercial fatigue tools implementing CZM are far more recent [21].

Despite this, there is currently a lack of VCCT-based fatigue crack growth models and relative case studies compared to CZM. A possible explanation to such scarcity was provided by Pirondi et al. [5]: the authors compared the predictive capabilities of both VCCT and CZMs for fatigue crack propagation under mode I, mode II and mixed mode loading. Interestingly, while both techniques were shown to lead to very similar results, fatigue simulations using VCCT required from 90% up to 1380% more time than their CZMs counterparts. In addition, the vast majority of the works available on the topic use the FE code Abaqus [22] to perform crack propagation fatigue simulations with the VCCT [5,18,20,23–26]. As it will be extensively explained later (see Section 2), the Abaqus dedicated crack propagation tool also poses an additional limitation: a single Paris-like propagation law can be used throughout the entire simulation. This is a severe limitation, since the Paris law coefficients and exponents vary significantly with the local mode-mixity [3,24]. As a result, the accuracy of the approach is questionable for complex cases, where local variations of the mode-mixity are present along the crack front [24].

The implementation of the VCCT for fatigue crack growth simulations in the Abaqus is done via the Direct Cyclic (DC) algorithm (described in Section 2.1) [22]. This algorithm tentatively simulates the

whole stabilised cycles at certain time intervals. Due to its aforementioned wide adoption in the literature [5,18,20,23–26], in this work it will be considered as the benchmark for VCCT-based fatigue crack growth simulation. The aim of this work is to present a novel 3D VCCT-based algorithm called Sequential-Static Fatigue (SSF). Differently from the DC algorithm, the SSF launches a series of static simulations to evaluate the state of the delaminated interface after several elapsed cycles. The cycles interval between each simulation is computed so that the crack is propagated by at least one element length. Section 2 of this work will present the basic principles of the VCCT and how it is implemented in Abaqus, while Section 3 will present its implementation in the SSF algorithm. Section 4 will present the case studies on which the SSF algorithm will be tested to evaluate its performance and limitations, and to compare it with the standard VCCT implemented in Abaqus. These case studies include:

- Standard specimens, to evaluate the SSF performance against the DC algorithm and to validate it against standard experimental cases;
- A non-standard specimen, that serves as an experimental validation case involving a complex geometry;
- A real-life structure, to prove its industrial applicability.

Finally, the results of these case studies will be shown and critically discussed in Section 5.

2 VCCT and its implementation in Abaqus

In all Abaqus simulations involving VCCT, modelling the debonding of a pair of surfaces requires the definition of two mating surfaces [22]. One of them is defined as a “master surface”, the other is defined as a “slave surface”. Both surfaces are used for computation reasons; however, the slave surface is also used to define a set of “bonded nodes”. The nodes of the slave surface belonging to this set define the bonded portion of the interface, while nodes not belonging to it define the debonded region. Crack propagation is thus modelled by releasing (i.e. removing from the bonded nodes set) nodes of the slave surface. Moreover, Abaqus computes all the interaction quantities at the nodes of the slave surface. For these reasons, the discussions in this work exclusively refer to nodes belonging to the slave surface.

Moreover, the expression “bonded nodes” will be used to refer to the nodes belonging to the set of bonded nodes.

2.1 Static simulations

VCCT is based on the assumption that the strain energy released when a crack is extended by a certain length δa is equal to the work done to close the same crack by the same amount δa. Consider Figure 1, which schematically depicts a 2D FE model of a crack tip. According to the previous definition, the mode I and mode II components of the strain energy release rate $\mathcal{G}_{\mathrm{I}}$ and $\mathcal{G}_{\mathrm{II}}$ at node 2 are defined as follow:

$$\mathcal{G}_{\mathrm{I}} = \frac{1}{2b\delta a} F_{y,2}(v_1 - v_{1^*}) \tag{1}$$

$$\mathcal{G}_{\mathrm{II}} = \frac{1}{2b\delta a} F_{x,2}(u_1 - u_{1^*}) \tag{2}$$

where b is the width of the analysed component, $F_{x,2}$ and $F_{y,2}$ are the horizontal and vertical forces at node 2 and u_1, u_{1^*}, v_1, v_{1^*} are the horizontal and vertical displacements of nodes 1 and 1*, respectively.

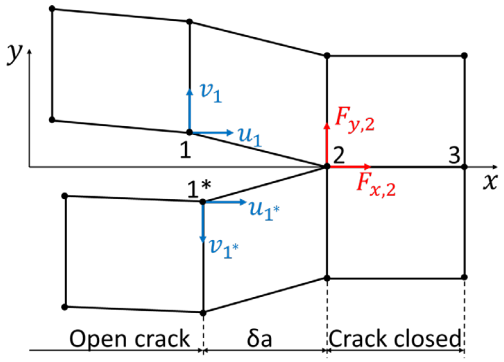

Figure 1: Schematic representation of the VCCT

The VCCT is implemented in Abaqus for the simulations of static crack propagation. In these simulations, node 2 will debond under mode I, creating 2 nodes, when:

$$\frac{\mathcal{G}_{\mathrm{I}}}{\mathcal{G}_{\mathrm{IC}}} \geq 1 \tag{3}$$

where $\mathcal{G}_{\mathrm{IC}}$ is the mode I fracture toughness. Expanding Equation 3 to the general case involving mode I, mode II and mode III, the criterion becomes:

$$\frac{\mathcal{G}_{\mathrm{eq}}}{\mathcal{G}_{\mathrm{C}}} \geq 1 \tag{4}$$

where $\mathcal{G}_{\mathrm{eq}}$ and $\mathcal{G}_{\mathrm{C}}$ are the equivalent strain energy release rate and fracture toughness, respectively. Among the different mode-mix formulae implemented in Abaqus, the one considered in this work is the BK law [27]. This is expressed by Equation 5 and Equation 6, where $\mathcal{G}_{\mathrm{I}}$, $\mathcal{G}_{\mathrm{II}}$ and $\mathcal{G}_{\mathrm{III}}$ are the mode I, mode II and mode III strain energy release rate, respectively, $\mathcal{G}_{\mathrm{IC}}$, $\mathcal{G}_{\mathrm{IIC}}$ and $\mathcal{G}_{\mathrm{IIIC}}$ are the mode I, mode II and mode III fracture toughness, respectively, and η is a fitting parameter.

$$\mathcal{G}_{\mathrm{eq}} = \mathcal{G}_{\mathrm{I}} + \mathcal{G}_{\mathrm{II}} + \mathcal{G}_{\mathrm{III}} \tag{5}$$

$$\mathcal{G}_{C} = \mathcal{G}_{\mathrm{IC}} + (\mathcal{G}_{\mathrm{IIC}} - \mathcal{G}_{\mathrm{IC}})\left(\frac{\mathcal{G}_{\mathrm{II}} + \mathcal{G}_{\mathrm{III}}}{\mathcal{G}_{\mathrm{I}} + \mathcal{G}_{\mathrm{II}} + \mathcal{G}_{\mathrm{III}}}\right)^{\eta} = \mathcal{G}_{\mathrm{IC}} + (\mathcal{G}_{\mathrm{IIC}} - \mathcal{G}_{\mathrm{IC}})(MM)^{\eta} \tag{6}$$

Note that the mode-mixity parameter MM was introduced in Equation 6.

2.2 Fatigue simulations with the DC algorithm

The capability of fatigue crack growth simulations is implemented in Abaqus via the DC algorithm. This algorithm aims to find a stabilised response of the analysed structure subjected to periodic loads at discrete time points (cycles). For this purpose, truncated Fourier series for the displacement solution $u(t)$ and residuals vector $\bar{R}(t)$ are used:

$$u(t) = u_0 + \sum_{k=1}^{n} (u_k^s \sin \mathrm{k}\omega\mathrm{t} + u_k^c \cos \mathrm{k}\omega\mathrm{t}) \tag{7}$$

$$\bar{R}(t) = R_0 + \sum_{k=1}^{n} (R_k^s \sin \mathrm{k}\omega\mathrm{t} + R_k^c \cos \mathrm{k}\omega\mathrm{t}) \tag{8}$$

where n is the number of terms in the Fourier series, $\omega = 2\pi/T$ is the angular frequency, u_o, u_k^s and u_k^c are the unknown displacement coefficients respectively associated to the residual vector coefficients R_o, R_k^s and R_k^c. Abaqus first obtains the residual vector $R(t)$ by using the standard Equation 9 of FE calculations, where $F(t)$ and $I(t)$ are the external and internal force vectors over the entire cycle. Equations 10-12 are then used to obtain the residual coefficients.

$$R(t) = F(t) - I(t) \tag{9}$$

$$R_0 = \frac{1}{T}\int_0^T R(t)dt \tag{10}$$

$$R_k^s = \frac{2}{T}\int_0^T R(t) \sin k\omega t \, dt \tag{11}$$

$$R_k^c = \frac{2}{T}\int_0^T R(t)\cos k\omega t\, dt \tag{12}$$

The displacement solution is obtained by solving for correction factors to the coefficients of the displacement Fourier function (Equation 7). An updated displacement solution can thus be obtained and used in the next iteration until convergence. The algorithm thus treats all quantities as time-dependent solutions over the entire loading cycle, which can thus be considered as single iterations. More details are provided in [22].

For fatigue crack growth problems in predefined interfaces (i.e. composites delamination or adhesive joints debonding), Paris law is used to determine the crack growth. In particular, Abaqus allows crack propagation if Equation 13 is satisfied, with c1 and c2 input coefficients, N is the cycle number and $\Delta\mathcal{G} = \mathcal{G}_{max} - \mathcal{G}_{min}$ is the fracture energy release rate interval between its maximum and minimum value. Once the propagation has started, Abaqus releases at least one node of the interface elements, thus extending the crack by a length a_N, by incrementing the cycle number of a quantity ΔN obtained via Equation 14 (Paris-like equation), where c_3 and c_4 are input parameters describing the Paris law.

$$\frac{N}{c_1\Delta\mathcal{G}^{c_2}} \geq 1.0 \tag{13}$$

$$\frac{da}{dN} = c_3\Delta\mathcal{G}^{c_4} \tag{14}$$

2.3 Main limitations of the DC algorithm

While the DC algorithm just described is an effective way to simulate fatigue crack propagations, it suffers from several limitations. The first major practical limitation is related to the computational efforts required by such simulation. As reported in the literature [5,23,28] and shown later in this work (see Section 5.3), the computational time required by some DC simulations can be prohibitively long also for simple structures.

Another important limitation of the described algorithm is related to the implementation of the crack propagation theories. As mentioned, a Paris-law based approach is adopted by Abaqus via Equation 14. However, Equation 14 also highlight that Abaqus considers a unique crack propagation law throughout the entire simulation, regardless of the local mode-mixity of the crack front. It is a well-known fact that this is rarely the case, since mode-mixity is a determining factor of the crack propagation response [29].

This limitation was already highlighted by Raimondo et al. [24] that used the DC algorithm for fatigue simulations of coupons and a composite stiffened panel. It was shown that, especially for the latter case, adopting a single set of Paris was likely the main source of inaccuracies. Indeed, several CZMs developed in the literature avoid this issue by considering a mode-mixity dependent coefficient and exponent of the Paris-law equation [3,4,8,29–31]. Moreover, Abaqus considers the fracture energy release rate interval $\Delta\mathcal{G}$ as the crack driving force in the Paris-law. However, this is still a debated subject in the literature, since crack growth rate can be related to several formulations of the energy release rate, according to the loading conditions [17,32].

Overall, the DC algorithm allows for fatigue crack propagation to be simulated in a commercial FE software. However, it is both computationally expensive and often of questionable accuracy due to inaccurate assumptions.

3 The Sequential Static Fatigue algorithm

The SSF algorithm is based on a series of static simulations instead of a single cyclic one. Each static simulation models a ramp from zero to the maximum load/displacement applied in the fatigue cycles. The objective is to extract the values of $\mathcal{G}$ at the bonded interface during peaks and troughs of the load cycles. This allows to take into account also the effects of the load ratio R. The cycle interval between each static simulation is computed via an external Python [33] master code, so that at least one node is removed from the bonded nodes set from one simulation to the next one. In this way, the crack advances by the distance between two adjacent nodes. This is repeated until a specific number of cycles is reached or the crack propagates through the entire bonded surface. The algorithm is schematically summarised in Figure 2; the SSF algorithm will be described following the blocks presented in Figure 2.

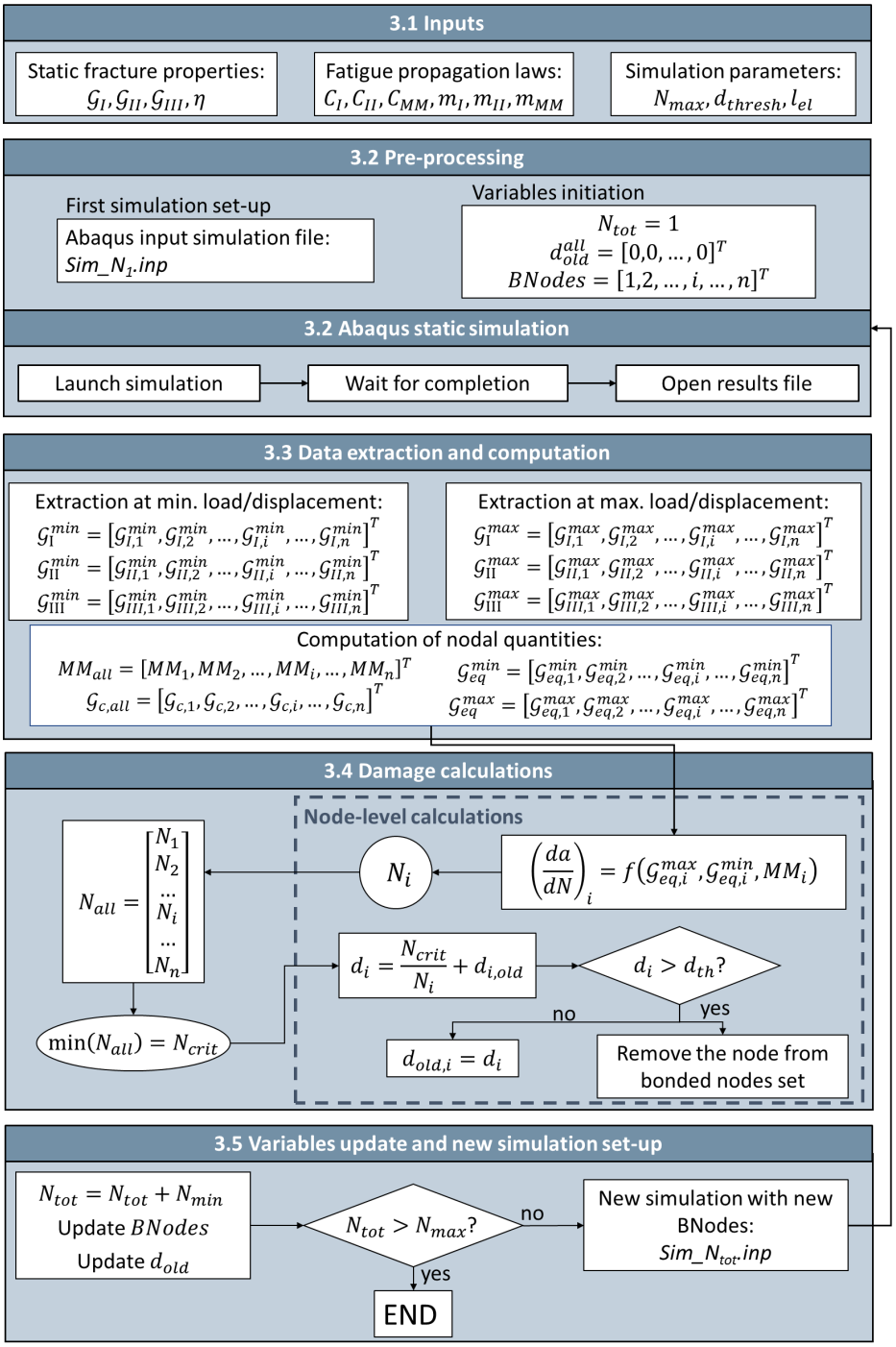

Figure 2: Implementation flowchart of the SSF algorithm

3.1 Inputs

The inputs required by the SSF algorithm are:

- The static fracture properties, namely a mode-mix formulation that describes the fracture toughness as a function of the local mode-mixity (for example, the B-K law of Equations 5-6);

- A fatigue propagation law, usually a Paris-like law. In this work, such law is expressed by Equations 16-18;

- Other simulation parameters, namely the maximum number of cycles to be simulated N_{max}, the damage threshold d_{th} above which nodes are released (see Section 3.4.2) and the average distance between adjacent nodes l_{el}.

3.2 Pre-processing and static simulation

A reference static Abaqus input file (.inp) is first created. This file describes the analysis to be performed in terms of geometry of the structure, material parameters, boundary conditions and initial bonded nodes set ("*BNodes*" in Figure 2). In all simulations, the structure is subjected to a ramp going from zero to the maximum applied load/displacement in the fatigue cycles to be simulated. Along this ramp, the FE simulation is forced also to compute the response of the structure at the minimum load/displacement applied in the fatigue cycles. In this way, information at both maximum and minimum applied load/displacement are available.

A Python master code then launches the first simulation, while keeping track on the total number of elapsed cycles N_{tot}; for the first simulation, N_{tot} is 1. While the fatigue calculations will be performed later by the Python master code, fracture toughness information are also available to Abaqus in each static simulation. This means that, if the applied load/displacement is high enough to propagate the crack statically (i.e. release bonded nodes statically), Abaqus automatically does so during the load ramp. To reproduce the very high propagation rate of the final stage of the Paris law, all static simulations following the first one have the local fracture toughness reduced by 10% [22]. This is also implemented in the DC algorithm, where nodes are released statically if they exceed 90% of their fracture toughness.

3.3 Data extraction and computation

Once one simulation is finished, the Python code autonomously opens the results file (.odb) and extracts the values of the $\mathcal{G}$ components on each remaining bonded node. These values are obtained for the increments corresponding to the minimum and maximum of the applied load/displacement in the fatigue cycles. Quantities extracted from all the n nodes are gathered into vectors:

- $\mathcal{G}_{\mathrm{I}}^{min}$: vector of all $\mathcal{G}_{\mathrm{I}}$ at the minimum load/displacement

- $\mathcal{G}_{\mathrm{II}}^{min}$: vector of all $\mathcal{G}_{\mathrm{II}}$ at the minimum load/displacement
- $\mathcal{G}_{\mathrm{III}}^{min}$: vector of all $\mathcal{G}_{\mathrm{III}}$ at the minimum load/displacement
- $\mathcal{G}_{\mathrm{I}}^{max}$: vector of all $\mathcal{G}_{\mathrm{I}}$ at the maximum load/displacement
- $\mathcal{G}_{\mathrm{II}}^{max}$: vector of all $\mathcal{G}_{\mathrm{II}}$ at the maximum load/displacement
- $\mathcal{G}_{\mathrm{III}}^{max}$: vector of all $\mathcal{G}_{\mathrm{III}}$ at the maximum load/displacement

These vectors are then used to compute additional vectors of nodal parameters:

- MM_{all}: vector of all mode-mixities at the maximum load/displacement
- $\mathcal{G}_{c,all}$: vector of all fracture toughnesses (see Equation 6)
- $\mathcal{G}_{eq}^{min}$: vector of the equivalent strain energy release rate at the minimum load/displacement
- $\mathcal{G}_{eq}^{max}$: vector of the equivalent strain energy release rate at the maximum load/displacement

Note that the outlined approach assumes that the minimum and maximum values of the $\mathcal{G}$ components occur at the minimum and maximum values of the globally applied load. This may not be true for very complex analyses involving real components, and a different approach may be required.

3.4 Damage calculations

3.4.1 Crack propagation law

The local crack propagation rate $(da/dN)_i$ can now be computed for each node, provided a suitable crack propagation law is given as input. Since fatigue crack growth calculations are performed by the Python algorithm, any crack propagation law of the form of Equation 15 can be implemented [3,4,31].

$$\left(\frac{da}{dN}\right)_i = f\left(\mathcal{G}_{eq,i}^{max}, \mathcal{G}_{eq,i}^{min}, MM_i\right) \tag{15}$$

Unless otherwise stated, following the indications of [3,31], a Paris-like propagation law was implemented in this work described as follows:

$$\frac{da}{dN} = C\left(\frac{\Delta\mathcal{G}}{\mathcal{G}_c}\right)^m \tag{16}$$

$$\ln C = \ln C_I + \ln C_m \cdot MM + \ln\left(\frac{C_{II}}{C_I C_m}\right) \cdot MM^2 \tag{17}$$

$$m = m_I + m_m \cdot MM + (m_{II} - m_I - m_m) \cdot MM^2 \tag{18}$$

where C and m are the Paris coefficient and exponent for the local mode-mixity MM, C_I and C_{II} are the Paris coefficients for pure mode I and mode II, respectively, m_I and m_{II} are the Paris exponents for pure mode I and mode II, respectively, and C_m and m_m are material fitting parameters. Note that the mode-mixity MM is calculated from the increment corresponding to the maximum applied load/displacement.

3.4.2 Node releasing criterion

Removing one node on the crack front from the bonded nodes set is equivalent to extending the crack by a length l_{el}, that is the average distance between adjacent nodes (for linear elements, this is the average finite element length). Therefore, Equation 15 can be expressed in a discrete form (see Equation 19). The numbers of cycles N_i required to remove each i^{th}-node from the bonded nodes set is thus calculated via Equation 20. While this approach may be inaccurate for irregular meshes with varying element size, it is commonly adopted for the vast majority of the CZMs [3–8] and VCCT models available in the literature, including the DC algorithm [19,20,22].

$$\frac{l_{el}}{\Delta N} = f(\mathcal{G}_{\max}, \mathcal{G}_{\min}, MM) \tag{19}$$

$$N_i = \frac{l_{el}}{f(\mathcal{G}_{i,max}, \mathcal{G}_{i,min}, MM_i)} \tag{20}$$

The nodes on the crack front will have the highest values of $\mathcal{G}_{eq}$, and will thus require less cycles to be removed. Therefore, the minimum number of cycles N_{crit} is obtained from this procedure, that is, the smallest number of cycles required to remove one node. The node associated to N_{crit} is thus the most critical one. To let the crack propagate, nodes have to be removed from the bonded nodes set. For this purpose, a damage variable d_i can be defined for each bonded node i as:

$$d_i = \frac{N_{crit}}{N_i} + d_{i,old} \tag{21}$$

$$d_i \geq 1 \tag{22}$$

where N_i is obtained via Equation 20. Moreover, $d_{i,old}$ is the damage variable of the same node computed in the previous simulation (zero for the first simulation). Updating the damage variable from one simulation to the other (via the $d_{i,old}$ term) allows to keep track of the total effect of the elapsed cycles; as shown in Section 5.2, this is critical to achieve accurate growth predictions. Equation 22 shows the

criterion by which nodes are removed from the bonded nodes set, that is, when the damage variable d_i becomes higher than 1.

3.5 Variables update and new simulation set-up

A new static simulation can be now performed by increasing the total cycle count N_{tot} by N_{crit} and by updating the bonded nodes set. As shown in Equation 21, the damage variable associated to each node is also updated, so that the variable also keeps track of the total effect of the elapsed cycles. The algorithm keeps launching static simulations and propagating the crack front until a maximum number of cycles N_{max} is reached or until there are no nodes left in the bonded nodes set (i.e., debonding of the entire surface).

3.6 SSF considerations

A significant advantage of the SSF algorithm compared to the DC one implemented in Abaqus can already be highlighted by comparing Equation 14 and Equations 15-18. The DC algorithm considers a single set of Paris parameters as input throughout the entire simulation. Any local variation of the mode-mixity is thus not considered. This is a critical point, since for real structures and even for some specimens, the local mode-mixity can vary significantly both along the crack front and during the loading history of the structure, due to the crack propagation [24]. Conversely, the SSF algorithm is able to deal with different local mode-mixities at once within the same simulation.

Finally, note that the Abaqus implementation of the VCCT considers $\Delta\mathcal{G}$ as the crack driving force for the fatigue crack propagation (see Equation 14). The SSF, instead, can use several functions of the strain energy release rate as crack driving force: in this case, following [3,31], $\Delta\mathcal{G}/\mathcal{G}_c$ was used (see Equation 16). However, the scientific literature on the subject has not yet identified a definitive crack driving force [17,32]. The SSF allows an easy variation of the implemented crack propagation law, thanks to its Python master code, whereas the DC implementation imposes the ΔG formulation to the user.

4 Case studies

4.1 DCB specimen under different mode-mixities

Several models developed in the literature were validated with crack growth tests performed on unidirectional carbon fibre reinforced epoxy Double Cantilever Beam (DCB) specimens loaded in mode

I, mode II and with 50% mixed mode [3,4,34]. The DCB specimen is shown in Figure 3a, with the relative dimensions reported in Table 1. The considered boundary conditions are shown in Figure 3b, Figure 3c and Figure 3d for the three considered cases, respectively. In particular, bending moments are applied to the DCB arms in all cases: they are equal and opposite for mode I, equal for mode II, and proportional for the mixed mode case. For the latter case, the simplified beam theory can be used to compute the ratio ρ between the two applied moments required to achieve a mode-mixity of 0.5 [3,34–36]:

$$\rho = \frac{1 - \frac{\sqrt{3}}{2}}{1 + \frac{\sqrt{3}}{2}} \tag{23}$$

Table 1: Geometrical details of the DCB specimen.

Length L	Initial debonding length a_0	Thickness t	Width
100 mm	35 mm	1.55 mm	20 mm

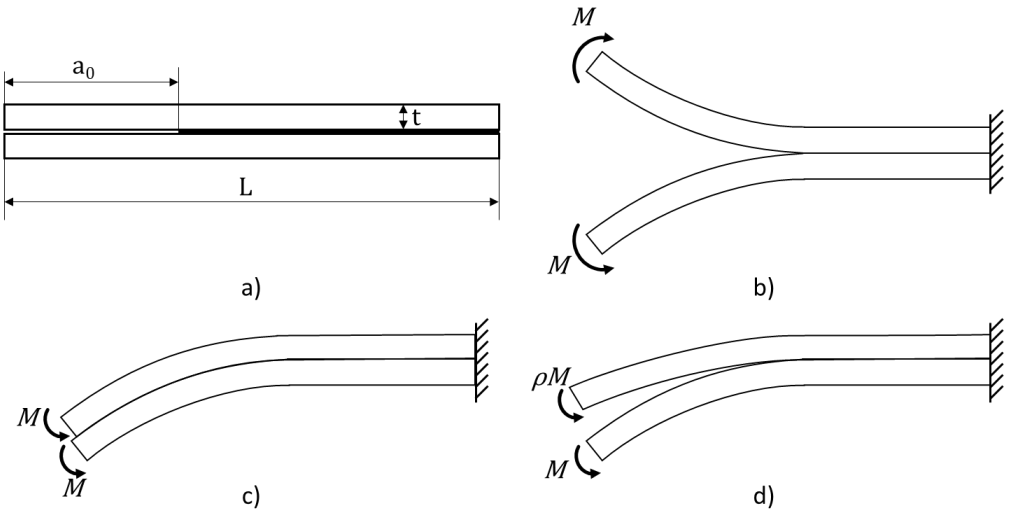

Figure 3: Carbon-epoxy DCB used for validation [3,4,34]: a) geometry and boundary conditions in b) mode I, c) mode II and d) mixed-mode.

Since a moment is applied to the tip of the DCB arms, the applied $\mathcal{G}$ does not depend on the crack length. As a result, the crack propagation rate is constant during the tests (and the simulations), while the crack propagates [3,31,34,36].

For each loading condition, four simulations were performed to test the SSF capabilities for a wide range of applied $\mathcal{G}$. Table 2 shows the considered moments in all cases. Moreover, the theoretical values of the applied $\mathcal{G}/\mathcal{G}_c$ are also reported. To compute the applied strain energy release rates for mode I, mode II and 0.5 mixed-mode conditions, Equation 24, Equation 25 and Equation 26 were used, respectively [36]. In the equations, M is the applied moment, E is the material's Young's modulus, I and B are the second moment of inertia and the width of the specimen arm, respectively. Finally, for the mixed-mode case, $\mathcal{G}_c$ was calculated via Equation 6.

$$G_I = \frac{M^2}{EIB} \tag{24}$$

$$G_{II} = \frac{3M^2}{4EIB} \tag{25}$$

$$G_I = G_{II} = \frac{3}{4\left(1+\frac{\sqrt{3}}{2}\right)^2}\frac{M^2}{EIB} \tag{26}$$

Table 2: Simulation cases used in the validation. Cases marked with "*" were used for the comparison with the benchmark (see Section 4.2).

Pure mode I MM=0	Mixed-mode MM=0.5	Pure mode II MM=1
M = 750 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.145	M = 1500 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.175	M = 2000 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.2
M = 1000 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.258	M = 2000 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.31	M = 2500 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.31
M = 1500 Nmm* $\mathcal{G}/\mathcal{G}_c$ = 0.581	M = 2500 Nmm* $\mathcal{G}/\mathcal{G}_c$ = 0.486	M = 3100 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.483
M = 1750 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.79	M= 3300 Nmm $\mathcal{G}/\mathcal{G}_c$ = 0.847	M = 3750 Nmm* $\mathcal{G}/\mathcal{G}_c$ = 0.70

The DCB numerical model adopted in the simulations is shown in Figure 4. To reduce the computational time of all the performed simulations, the crack was allowed to propagate for 30 mm only. For this reason, the whole specimen was separated in three regions: the initially debonded area, the crack propagation area and an area where the adherends are perfectly bonded to each other. A preliminary mesh sensitivity analysis was first conducted, whose result were reported in Section 5.1. Based on this analysis,

a 0.5 mm mesh seed was used for the crack propagation area, since it is the region of highest interest, while a mesh seed of 1 mm was used in the other two regions. A correct use of the VCCT in Abaqus requires an initial crack tip in the debonding surface: this is created by removing some nodes that lay on the propagation surface from the bonded nodes set [22]. To accurately model the geometry defined in Table 1, the initial debonded area was set to 34 mm; two rows of nodes of the crack propagation area were removed from the bonded nodes set (see Figure 4), to extend the initial debonding by an additional 1 mm. In this way, the initial debonding of 35 mm was correctly obtained. The crack propagation area is thus 31 mm long, with only 30 mm available for the actual crack propagation. The remaining 35 mm of the DCB arms were tied together.

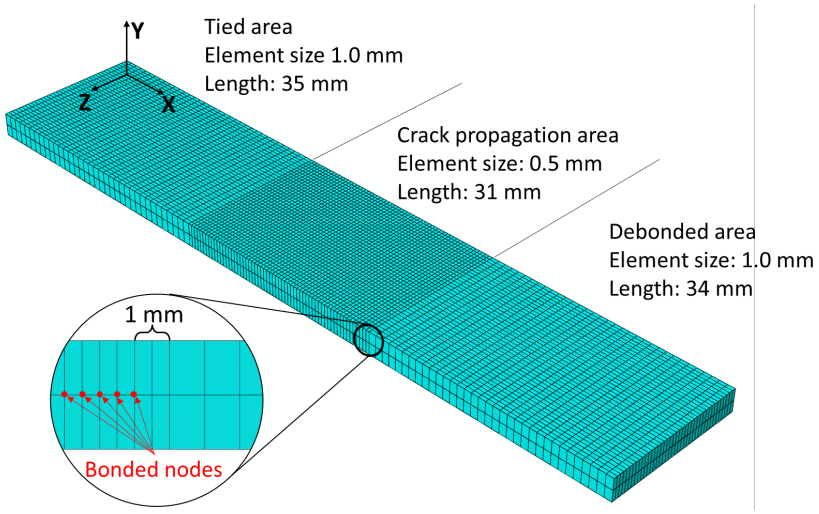

Figure 4: FE model of the analysed DCB

Reduced integration 8-nodes continuum shell elements SC8R were used, with one element through the thickness. The two arms were modelled as a composite laminate, using the dedicated Abaqus tool [22], with stacking sequence $[0°]_{12}$. The material properties used are reported in Table 3 [3,31,34]. The R-ratio of the experiments was 0.1 [34]. Finally, l_{el} was set to be equal to 0.5 mm and the maximum number of cycles N_{max} was set to 10^{12} cycles, since the SSF algorithm automatically stopped after the 30 mm delamination occurred.

Table 3: Material properties used in the DCB simulations [3,31,34]

Laminate properties	Interface properties	Propagation properties

E11	120	[GPa]	$\mathcal{G}_{Ic}$	0.266	[N/mm]	C_I	3.08e-3	[mm/cycle]
E22=E33	10.5	[GPa]	$\mathcal{G}_{IIc}$	1.002	[N/mm]	C_{II}	0.149	[mm/cycle]
G12=G13	5.3	[GPa]	$\mathcal{G}_{IIIc}$	1.002	[N/mm]	C_m	458087	[mm/cycle]
G23	3.5	[GPa]	η	2.73	[-]	m_I	5.4	[-]
ν12= ν13	0.3	[-]				m_{II}	4.5	[-]
ν23	0.51	[-]				m_m	4.94	[-]

4.2 Performance comparison between the DC and the SSF algorithms

The SSF algorithm was compared to the DC one in terms of accuracy and computational efficiency. To this end, the same DCB specimen described in Section 4.1 was used. However, due to the different Paris laws implemented (compare Equation 14 and Equation 16) and the prohibitively large times required by the DC simulations, some modifications were needed.

First of all, DC parameters c_1 and c_2 were taken so that propagation occurs immediately, as often done in the literature [10]; moreover, parameters c_3 and c_4 were obtained via Equation 27 and 28, derived from Equation 14 and Equation 16. The adopted parameters are listed in Table 4; the remaining material parameters are those listed in Table 3.

$$c_3 = \frac{C}{G_c^m} \tag{27}$$

$$c_4 = m \tag{28}$$

Table 4: DC parameters.

Parameter	Mode I	Mixed Mode	Mode II
c_1 [N/mm·cycle]	0.5	0.5	0.5
c_2 [-]]	-0.1	-0.1	-0.1
c_3 [mm^2/N·cycle]	4.44	0.14767	120.5
c_4 [-]	5.4	6.41	4.5

A single mode I simulation was performed with the mesh described in Figure 4. The simulation was stopped once an entire row of bonded nodes was released, that is when the crack propagated by 0.5 mm. For this simulation, the applied moment was 1750 Nmm.

In addition, the mesh was modified to perform additional simulations with a longer propagation. Considering Figure 4, the crack propagation region was reduced to 10 mm, and a 1 mm mesh size was used for the entire specimen. As it will be shown in Section 4.1, the use of such coarse mesh leads to inaccurate evaluation of the applied $\mathcal{G}$. However, this is not a concern for the comparison of the DC and the SSF, since the same error is done in both algorithms. The simulated cases are the ones highlighted in Table 2 by an asterisk "*".

To allow an accurate prediction of the stabilised cycle by the DC algorithm, it is necessary to simulate the sinusoidal shape of the load cycles. Therefore, before the actual DC step, a preliminary static step was simulated with a time-step of 1 second. A load ramp was simulated from the unloaded conditions to maximum load in the first 0.8 seconds; then, the load was decreased to its mean value during the fatigue history in the following 0.1 second; finally, it was held constant for the last 0.1 second so that the DC step could start from the mean load. This initial ramp was performed to assess the accuracy of the DC on the evaluation of $\mathcal{G}$ against the static general FE algorithm of Abaqus.

All simulations described in this section were performed on a workstation equipped with an Intel i7-8700K processor and 64 GB of RAM and using 10 cores; Abaqus 2021 was used for all cases [22].

4.3 Reinforced DCB

To perform an experimental validation on a non-standard specimen, the SSF algorithm was tested against the experimental study published by Carreras et al. [37]. The considered specimen is a DCB with additional bonded composite plates to act as local reinforcement, as schematically shown in Figure 5a [37]. Both DCB specimen arms and reinforcing plate were made of 8 unidirectional plies of carbon fibre/epoxy prepreg stacked at 0°. A 5 mm displacement was first applied quasi-statically, which already triggered some delamination. The specimen was subjected to 420000 cycles with 5 mm of maximum opening and with a load ratio of R=0.1. After these cycles, the applied displacement was increased quasi-

statically to 10 mm. Finally, the specimen was subjected to 10000 cycles at a 10 mm maximum opening with a load ratio of R=0.1.

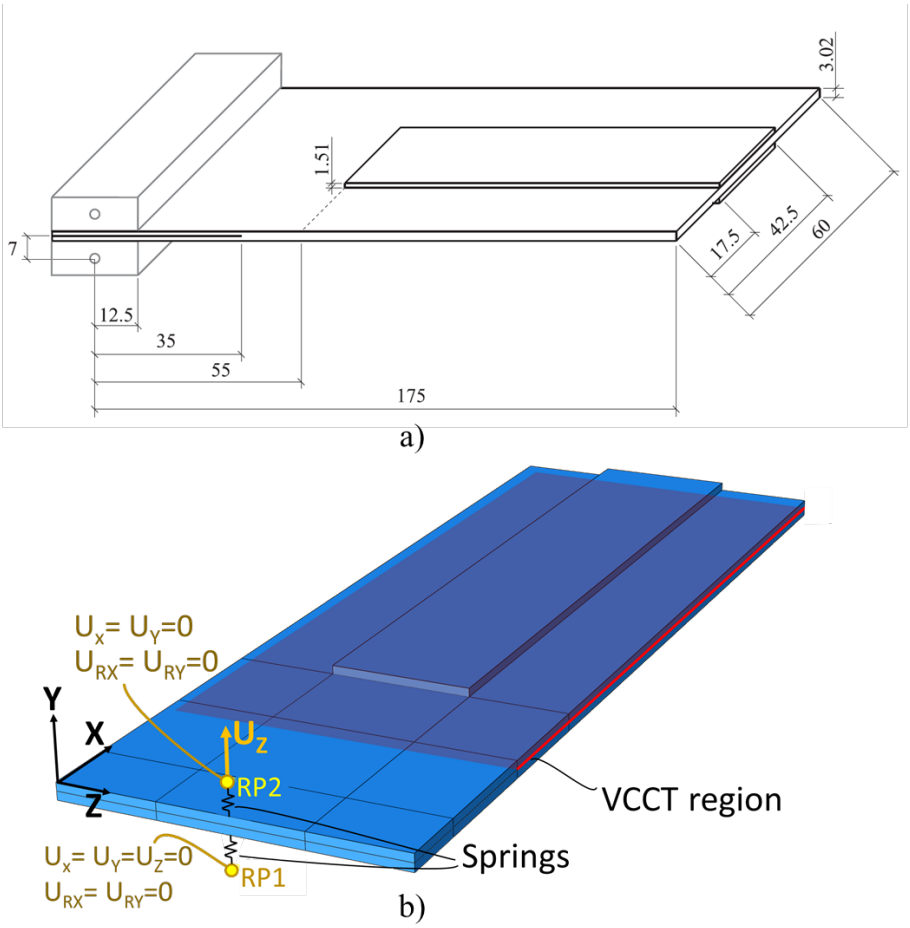

Figure 5: Reinforced DCB specimen: a) geometry taken from [38] and b) FE model.

The equivalent FE model is reported in Figure 5b, with its boundary conditions. As shown, the applied displacement was introduced via a spring with elastic constant K=1200 N/mm. This was done to compensate for an initial stiffness overprediction by the FE model. This problem and the adopted solution were also reported by Raimondo et al. [39]: the authors suggested that such stiffness mismatch is likely due to the compliance of the texting machine, which is not modelled in FE simulations. Continuum shell elements (SC8R), with one element per thickness, were used for both DCB arms and reinforcing plate, in combination with the composite laminate tool of Abaqus. The adopted material and interface properties are reported in Table 5, reported or calculated from the published data in [38]. For this study case, the adopted Paris law is reported in Equation 29, as done in the reference publication [38].

$$\frac{da}{dN} = C\mathcal{G}_{\max}^{m} \tag{27}$$

Table 5: Material properties published or calculated from [38].

Laminate properties			Interface properties			Propagation properties		
E11	154	[GPa]	$\mathcal{G}_{Ic}$	0.305	[N/mm]	C_I	564	[mm/cycle]
E22=E33	8.5	[GPa]	$\mathcal{G}_{IIc}$	2.77	[N/mm]	C_{II}	0.0121	[mm/cycle]
G12=G13	4.2	[GPa]	$\mathcal{G}_{IIIc}$	8.77	[N/mm]	C_m	$1.2 \cdot 10^{-6}$	[mm/cycle]
G23	3.036	[GPa]	η	2.0866	[-]	m_I	8.39	[-]
ν12= ν13	0.35	[-]				m_{II}	3.62	[-]
ν23	0.4	[-]				m_m	4.99	[-]

4.4 Single-stringer compression specimen

The SSF algorithm was also used on a more complex application, to show its relevance for industrial purposes. Raimondo et al. [24] tested the DC algorithm on a single-stringer composite panel previously manufactured and tested [26,40]. The authors highlighted the significant drawback of the DC algorithm to allow a single set of Paris parameters. The fatigue crack growth results obtained were thus questioned, and the authors highlighted the importance of solving this issue for complex industrial components. The same component is thus used, in this work, as industrially relevant study case for the SSF algorithm.

The panel geometry is reported in Figure 6. The panel is composed by a skin and a stringer, co-cured together. A 40 mm Teflon insert was used to serve as initial delamination. Both skin and panel were made of unidirectional plies of carbon-epoxy IM7/8552, and their stacking sequences were $[45/90/-45/0]_s$ and [-45/0/45/0/45/0/-45], respectively. The material parameters are reported in Table 6 [24].

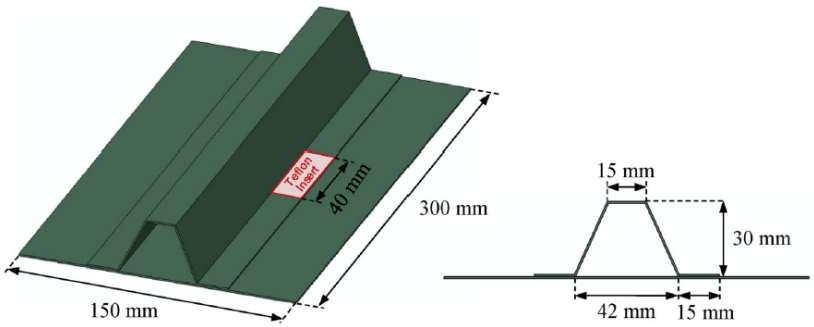

Figure 6: Composite panel geometry, from [24].

Table 6: IM7/8552 material properties [24].

Laminate properties			Interface properties		
E11	150	[GPa]	$\mathcal{G}_{Ic}$	0.277	[N/mm]
E22=E33	9.08	[GPa]	$\mathcal{G}_{IIc}$	0.788	[N/mm]
G12=G13	5.29	[GPa]	$\mathcal{G}_{IIIc}$	0.788	[N/mm]
G23	3.4	[GPa]	η	1.63	[-]
ν12= ν13	0.32	[-]			
ν23	0.45	[-]			

Regarding the fatigue simulations with the SSF algorithm, the applied boundary conditions are reported in Figure 7. The front and back surfaces of the panel were constrained to reference points, one of which was fixed; a 13.8 kN compressive load was applied to the other one. Regions spanning for 30 mm from these surfaces were prevented to move along the Y and Z axis, simulating the gripping regions of the real component. These are the same as those applied by Raimondo et al. [24]. However, the Paris parameters used were those reported in Table 3. This is because a consistent set of Paris parameters for the IM7/8552 material, with their dependency on the local mode-mixity, was not available in the literature, to the best of our knowledge. While this prevents a quantitative comparison between the SSF algorithm and both the DC and the experimental data of [24], it does not prevent a quantitative evaluation of the SSF capabilities.

A 2.5 mm and a 1 mm mesh were used for the skin and stringer, respectively; continuum shell elements (SC8R) were used for both components, in combination with the composite laminate tool of Abaqus. To avoid convergence issues through the buckling bifurcation point, geometrical imperfections were introduced as 1% of the displacements of the first buckling mode. This was also done by Raimondo et al. [24]. A preliminary buckling analysis was performed for this purpose. As shown in Figure 7, two crack growth regions were created on the right bonded area, while a normal contact only area was created in the teflon region. The remaining bonded surfaces of the two components were tied together. The maximum number of cycles N_{max} was set to ten millions cycles.

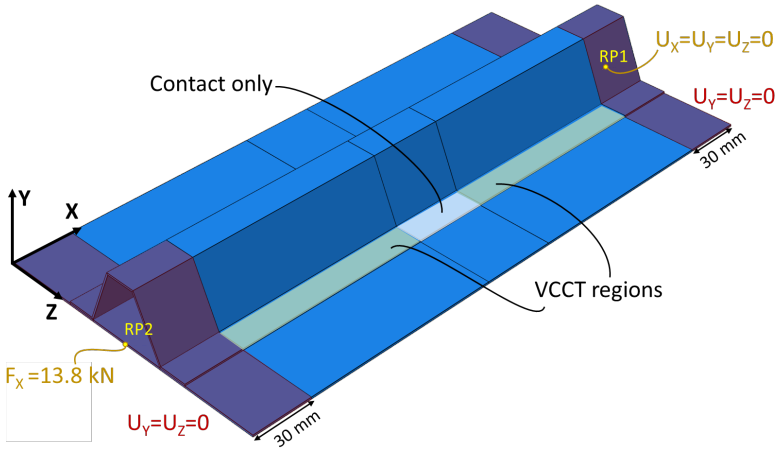

Figure 7: Boundary conditions on the composite panel simulations.

5 Results and discussion

5.1 Preliminary mesh sensitivity analysis

The VCCT is known to be a mesh-sensitive technique [41]. Therefore, a preliminary investigation was performed to evaluate the largest mesh size that leads to reasonably accurate results. For the crack propagation region of the specimens (see Figure 4), mesh sizes of 0.1 mm, 0.5 mm, 0.75 mm and 1 mm were tested under mode I conditions. The resulting distribution of $\mathcal{G}_I$ was then compared with the theoretical value obtained with Equation 24. Figure 8 shows the distribution of $\mathcal{G}_I$ along the width of the specimen. As shown, mesh sizes of 1 mm and 0.75 mm were too coarse to lead to accurate results. Conversely, both 0.5 mm and 0.1 mm mesh sizes lead to accurate predictions of $\mathcal{G}_I$. Discrepancies are present due to the 3D nature of the specimen, with $\mathcal{G}_I$ being higher at the centre and lower at the sides.

Nevertheless, the average values (shown as dashed lines in Figure 8) predicted by the two finest meshes are very close to the theoretical one.

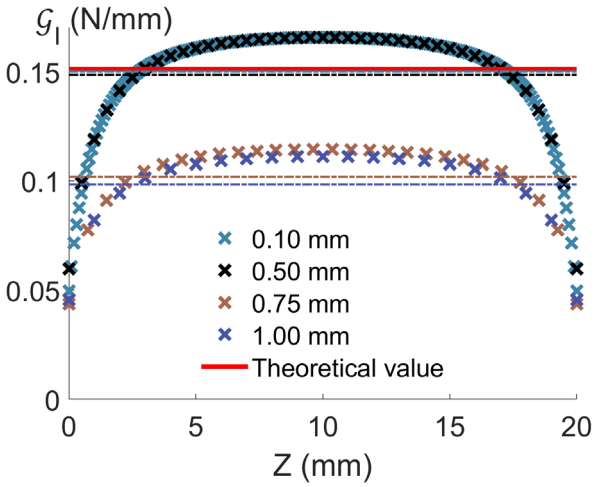

Figure 8: Mesh sensitivity results for mode I DCB. Dashed lines indicate average values.

The 0.5 mm mesh seemed to be the best choice in terms of accuracy and computational time required (the finer 0.1 mm mesh would require significantly more computational time). Therefore, its accuracy was also tested under mode II and mixed-mode conditions with MM=0.5. The theoretical values for these cases are obtained via Equation 25 and Equation 26, respectively.

Figure 9 shows that the 0.5 mm is able to accurately predict the applied $\mathcal{G}$ also under pure mode II (see Figure 9a) and under mixed-mode conditions (see Figure 9b). As a conclusion, the 0.5 mm mesh size was selected for all the simulations described in Section 4.1.

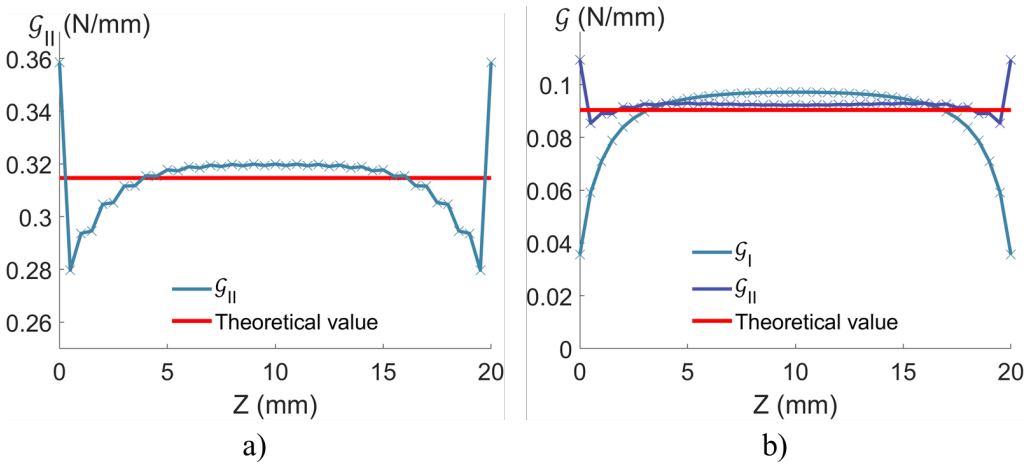

Figure 9: Predictions of $\mathcal{G}$ the 0.5 mm mesh for a) mode II and b) mixed-mode conditions

Moreover, this analysis also allowed to evaluate the 3D response of the DCB specimens under the different loading conditions. In particular, the 3D distribution of $\mathcal{G}$ shows that, when these components are present, $\mathcal{G}_{\mathrm{I}}$ and $\mathcal{G}_{\mathrm{II}}$ present absolute minima and maxima at the sides, respectively.

5.2 SSF performance

Figure 10 shows the results of the SSF algorithm in terms of crack growth rates. The abscissa of each SSF point is obtained considering the theoretical $\mathcal{G}$, while the ordinate is obtained as 30 mm divided by the total number of cycles required for the full propagation of the crack propagation region. Moreover, both experiments from Asp et al. [34] and the relative Paris curves are also plotted.

As shown, the predicted crack growth rate is in very good agreement with the input Paris laws. A slight overprediction of the crack growth rate is due to the higher local values of $\mathcal{G}$ along the crack front: these are shown in both Figure 8, Figure 9a and Figure 9b and for pure mode I, pure mode II and 0.5 mixed-mode case.

It is worth highlighting the effect of the cumulative damage formulation expressed in Equation 6. The term d^{old} allows to take into account the correct load history each node is subjected to, which is critical in 3D simulations. To prove this point, the simulations reported in Table 2 were also performed by removing the term from d^{old} Equation 6. In this way, a non-cumulative damage law is considered. The results of these simulations are also reported in Figure 10. Pure mode I and 0.5 mixed mode simulations show very small differences, which become negligible at higher values of $\mathcal{G}/\mathcal{G}_{\mathrm{c}}$. However, a significantly slower crack propagation was predicted for the mode II simulations. This was caused by higher fluctuations on the distribution of $\mathcal{G}_{\mathrm{II}}$ along the crack front when few nodes were released. As a result, many cycles were spent for the propagation of the crack front in the Z-direction (see Figure 4) before it could propagate in the negative X-direction. Considering, instead, a cumulative damage proved to correctly distribute the damage between nodes given the actual load history of each node. As a result, a more accurate propagation rate was predicted.

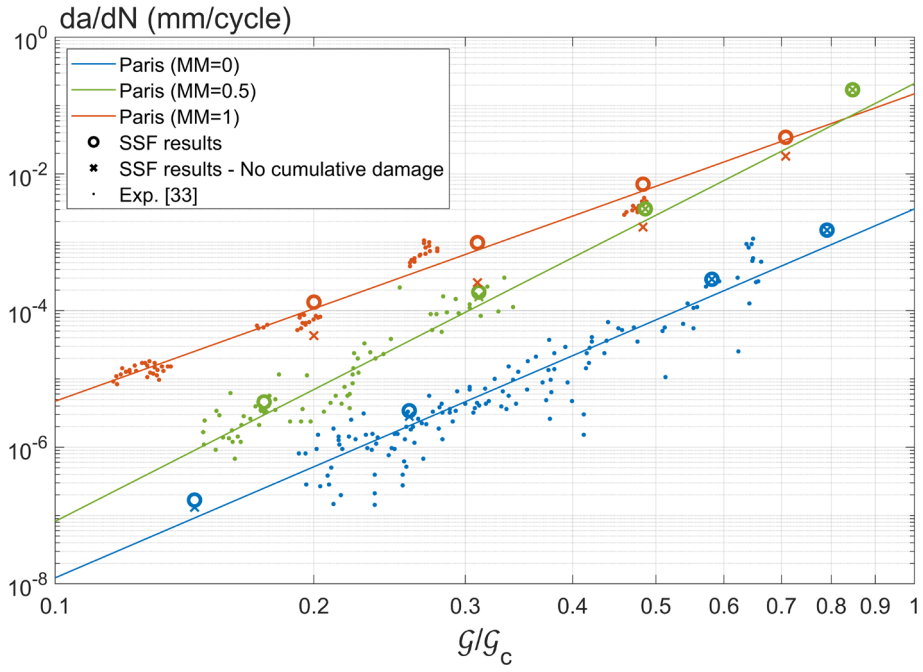

Figure 10: Crack growth rates predicted by the SSF algorithm

5.3 Comparison between SSF and DC

5.3.1 Releasing one row of nodes

The capabilities of both tested algorithms to release one row of nodes, corresponding to 0.5 mm propagation, were first assessed (as mentioned in Section 4.1). The DC algorithm and the SSF algorithm predicted a complete row debonding after 313 cycles and 353 cycles, respectively. Considering that both are based on the same VCCT algorithm, and have the same Paris parameters for pure mode I propagation, this difference was unexpected.

The reason for such difference is explained by Figure 11, that shows the global applied moment and the local value of $\mathcal{G}_I$ on the first released node in the first two unit-time intervals of the DC simulation. The first unit-time interval of the simulation, the load ramp described in Section 4.1, is simulated via a static general step; then, the DC algorithm is used for the cyclic load. As shown, despite the applied maximum moment is the same, the DC algorithm computes a higher local $\mathcal{G}_I$ at the same node. A reason for such behaviour was found in [19]: the DC algorithm discretises the load cycle in several time points; the default number of time points is 100, which was observed to introduce small errors ranging from 0.5% to 5%. In the case of this work, the difference between the two peaks of $\mathcal{G}_I$ is smaller than 3%; however, the power-law nature of Equation 14 makes this small inaccuracy the cause of significant differences in the

computed number of cycles. Conversely, the SSF algorithm relies exclusively on static general simulations, thus adopting the smaller value of $\mathcal{G}_I$ evaluated at the end of the static general step. As a result, the SSF leads to more accurate predictions than its counterpart, which tends to overestimate the crack propagation. It would be possible to improve the accuracy of the DC algorithm by increasing the number of time points; this, however, was already reported to significantly increase the computation time of the simulations [19]. Due to the already large computational time required by the simulations performed in this work, it was decided to leave the default value of the time points unaltered.

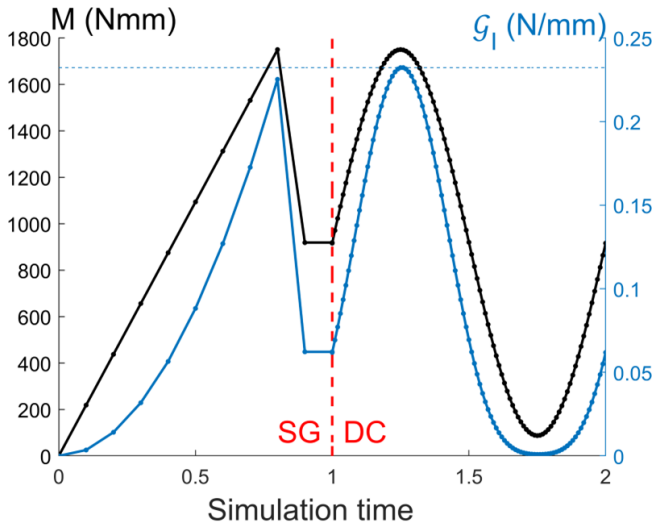

Figure 11: Applied moment and local value of of $\mathcal{G}_I$ on first released node in the first two unit-time intervals of the simulation implementing the DC algorithm. "SG" indicates "Static General" step. The dashed horizontal line is drawn at the second peak of $\mathcal{G}_I$.

The simulation implementing the DC algorithm required about 39 hours, 58 minutes and 54 seconds. The SSF algorithm, on the other hand, required only 2 minutes and 46 seconds; the row of nodes was released within the first two simulations. For this case, the overall simulation time was reduced by a factor of 867, i.e. by about 3 orders of magnitude.

Finally, the presented comparison was performed using the same 0.5 mm mesh, as mentioned. Longer computational times are required for finer meshes. For example, consider the DCB static simulation described in Section 5.1: the time required by the SSF simulation with a 0.1 mm mesh size was increased by a factor of 13 with respect to the simulation with the 0.5 mm mesh. Considering that the SSF is composed of several static simulations, it can be assumed that this time increase would be similar for SSF fatigue simulations of the two meshes. Therefore, SSF simulations on a 0.1 mm mesh size are likely to be

still significantly faster than DC simulations on a 0.5 mm mesh size. In addition, the time required by the DC algorithm likely depends on the minimum mesh size as well. Therefore, the SSF algorithm should be generally faster than the DC algorithm regardless the mesh size considered, although a direct comparison based on the mesh dependency of the two techniques is still required to confirm this.

5.3.2 Full propagations on a coarser mesh

Figure 12 shows the predicted crack growth (a-N) by the SSF and the DC algorithms for the different loading conditions. The curves were extracted at the most critical locations, namely the central nodes for mode I and mixed-mode, and the side nodes for mode II (see Figure 8 and Figure 9a). As shown, the DC always predicts a faster propagation; the difference is reduced for the pure mode II simulation (see Figure 12c). Such difference is the result of the aforementioned overestimation of the applied $\mathcal{G}$ in the DC step. Nevertheless, in all tested cases, the difference is not very significant considering the high scatter observed in fatigue-driven delaminations.

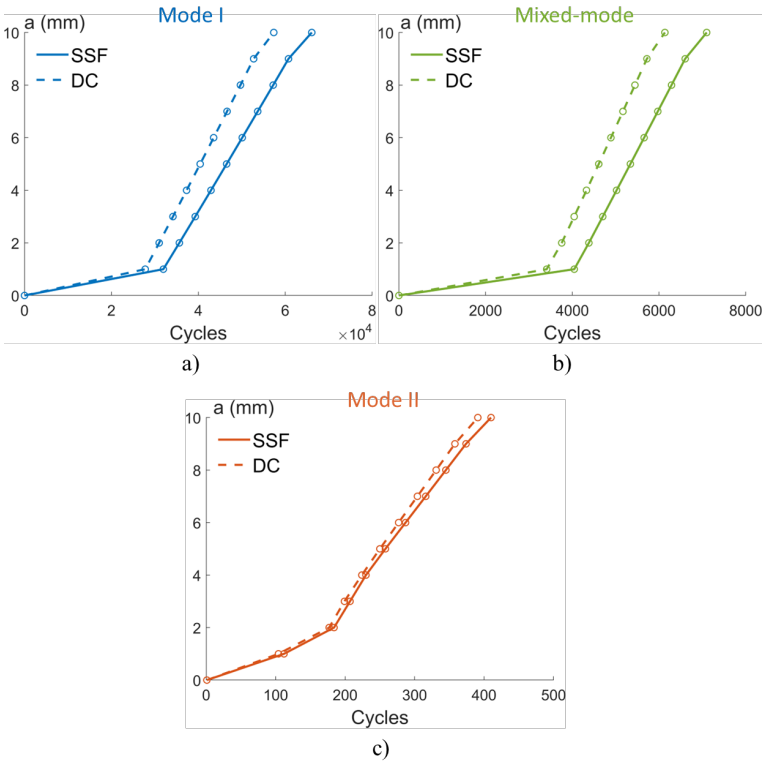

Figure 12: Predicted growth by the SSF and the DC algorithms under a) mode I, b) mixed-mode and c) mode II loading conditions.

Table 7 reports the simulation times required by both algorithms for all cases, together with the ratio of the DC time over the SSF one (the previous case of the single row of nodes delamination is also included, for the sake of completeness). As shown, the SSF proved to be significantly more computationally efficient than the DC algorithm in all cases. The time required by the SSF algorithm was between two and three orders of magnitude shorter than the DC algorithm in all cases.

Table 7: Time comparison between the DC and the SSF algorithms

Simulation case	DC time	SSF time	Time ratio
Fine mesh – mode I 0.5 mm propagation	39 h, 58 m, 54 s	2 m, 46 s	867
Coarse mesh – mode I 10 mm propagation	86 h, 42 m, 45 s	8 m, 29 s	613
Coarse mesh – mixed-mode 10 mm propagation	76 h, 3 m, 15 s	9 m, 46 s	305
Coarse mesh – mode II 10 mm propagation	52 h, 43 m, 48 s	14 m, 56 s	212

Overall, the SSF algorithm:

- drastically reduced the computational time of fatigue delamination growth simulations;
- was able to achieve a higher or equal accuracy than the DC algorithm, since it is based on static general simulations, more accurate than the DC ones.

Finally, it is worth highlighting that Pirondi et al. [5] reported time reduction factors ranging from 2 to 15 when adopting a CZM instead of VCCT-based DC. While a direct comparison between CZMs and the SSF algorithm is not performed in this work, this preliminary indirect comparison suggests that the SSF is significantly more computationally efficient than the CZM developed in [5] as well.

5.4 Reinforced DCB

Figure 13 shows the performance of the SSF algorithm against the reinforced DCB published in [38]. As shown, predictions are quite accurate for the first fatigue step (420000 cycles with 5 mm opening

displacement): the predicted applied force within 5% of the experimental one (Figure 13a). Conversely, a significant overestimation of the stiffness loss is present in the second fatigue step (10000 cycles with 10 mm opening displacement, Figure 13b). The crack location and shape are closer to the experimental ones for the initial phases of the first load ramp and fatigue step (see Figure 14a-e) than in the last phases (see Figure 14f-h); coherently, crack propagation is instead significantly overpredicted for the second load ramp and fatigue step (see Figure 15a-e).

This behaviour is likely due to the edges of the crack becoming oblique when this approaches the reinforcement arms. The discontinuous nature of the mesh causes higher local values of the SERR when the crack moves from one row of nodes to the other, as shown in the example of Figure 16. It was verified that a finer mesh, with a 0.1 mm element size, would reduce these SERR values by about 30%. This would likely cause a significant prediction improvement of the crack propagation. However, such mesh size would also cause a significant increase in the computational time required by these simulations; high performance computing clusters could be required, as done in [39] for a similar mesh. Therefore, the observed discrepancies are not due to an SSF specific problem, but are instead related to the mesh sensitivity of VCCT. This is proven by the inaccurate crack propagation predictions of the simple static simulation of the second static ramp (Figure 15a), where the SSF algorithm was not yet used. Therefore, considering the fair predictions of the SSF for the first fatigue step, it was decided to not perform additional simulations with a finer mesh.

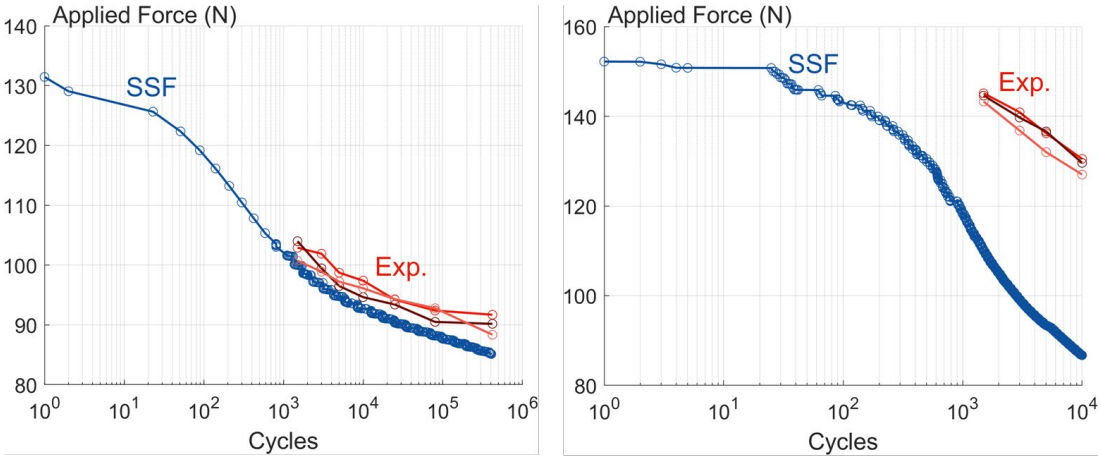

Figure 13: Reinforced DCB load-displacements curves in the a) first fatigue step and b) second fatigue step.

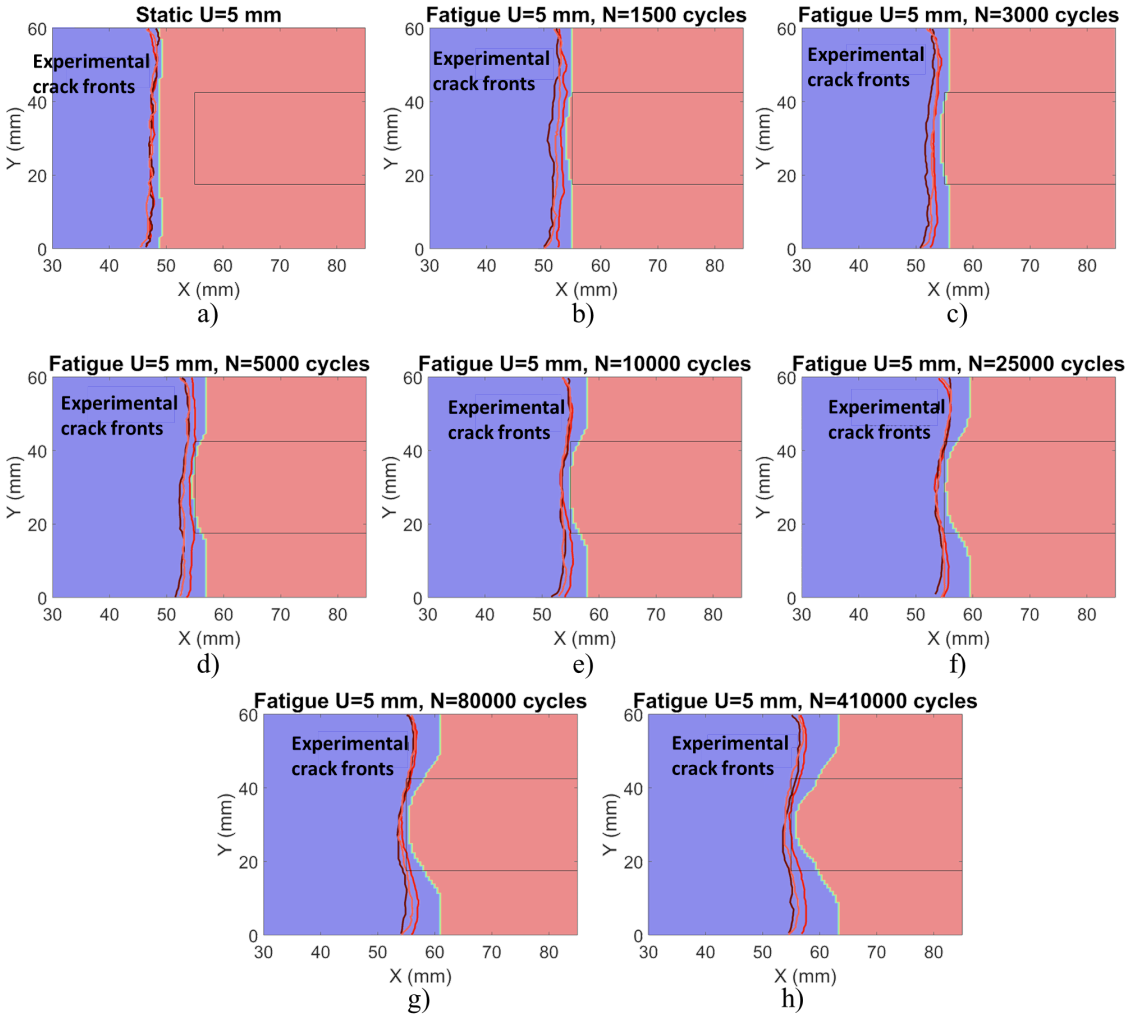

Figure 14: Experimental crack fronts (red lines) against numerical predictions (red area is bonded, blue area is debonded) for a) the first static ramp and b-h) the first fatigue step.

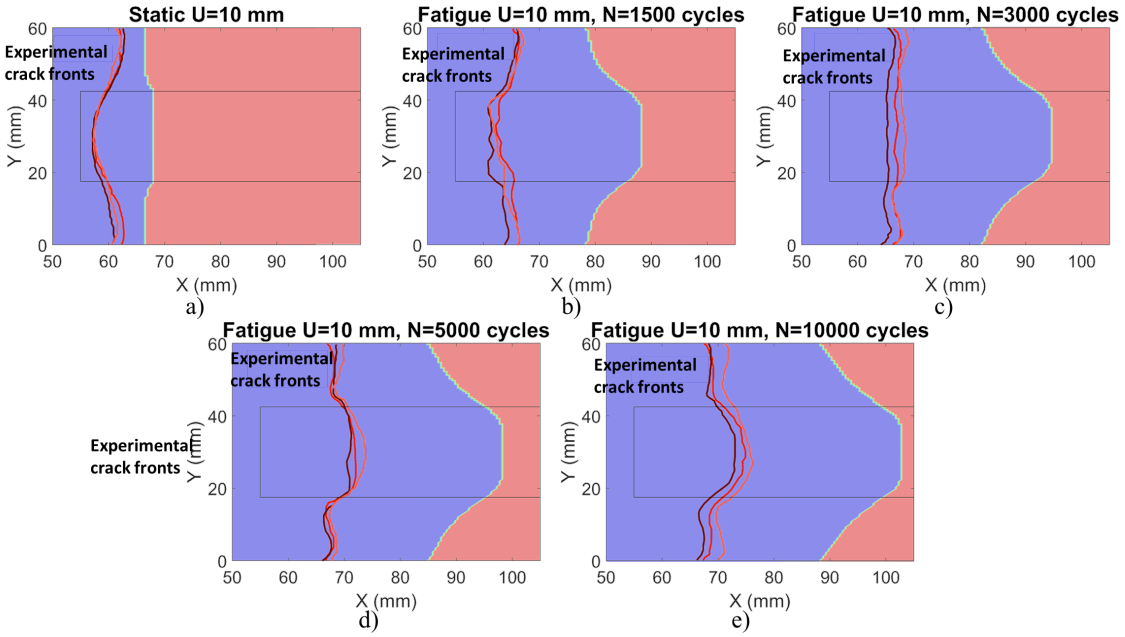

Figure 15: Experimental crack fronts (red lines) against numerical predictions (red area is bonded, blue area is debonded) for a) the second static ramp and b-e) the second fatigue step.

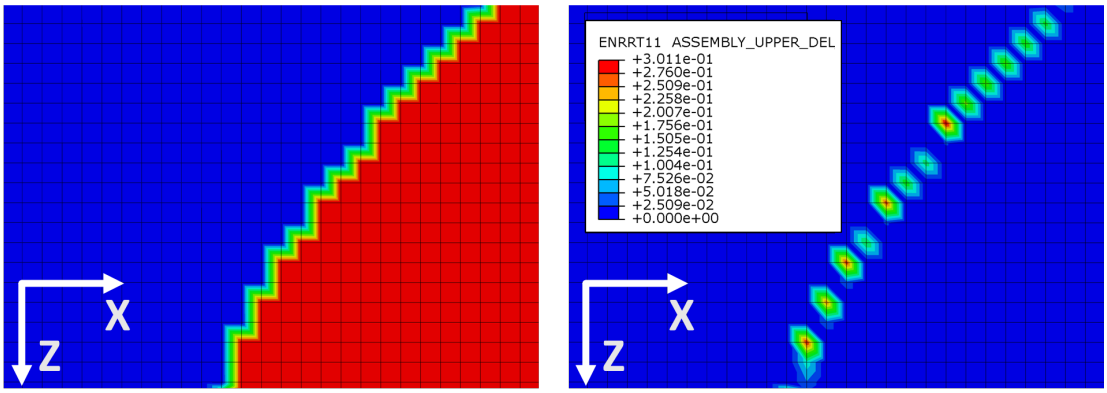

Figure 16: a) A detail of the oblique crack edge in the 1500 cycles simulation of the second fatigue step (red area is bonded, blue area is debonded) and b) the local higher values of the SERR caused by the mesh (mode I SERR only is shown as this was the main component of this case).

5.5 Single-stringer compression specimen

Figure 17 shows the evolution of the delamination in the composite panel. As shown, the propagation is non-symmetric: on the right side, delamination starts from the outer edge, while the opposite occurs in the left side. This is due to the introduced imperfection, as also reported in [24]. Propagation occurs steadily until about 1 million and 700 thousand cycles, after which the SSF algorithm automatically stopped. This is because releasing more nodes from the delaminated defect shown in Figure 17d would require about 26 million more cycles. While, as mentioned, the input uncertainties prevent a direct experimental comparison, it is worth highlighting the capabilities of the SSF algorithm. Considering the relatively low

applied load, the algorithm predicted a stable propagation until a maximum delamination of 14 mm and 7 mm on the left and right side, respectively. Such propagation considered the effect of the local mode-mixity, unlike what it is possible in the DC algorithm [24]. Moreover, the entire procedure required about 8 hours, 3 minutes and 5 seconds using 10 CPU cores. Raimondo et al. [24] reported 97 hours of simulation time for a 13.4 mm of total delamination increase (averaged over the width), using 20 CPU cores. Overall, the SSF proved to be more accurate and computationally more efficient than the DC algorithm also in case of industrially relevant cases.

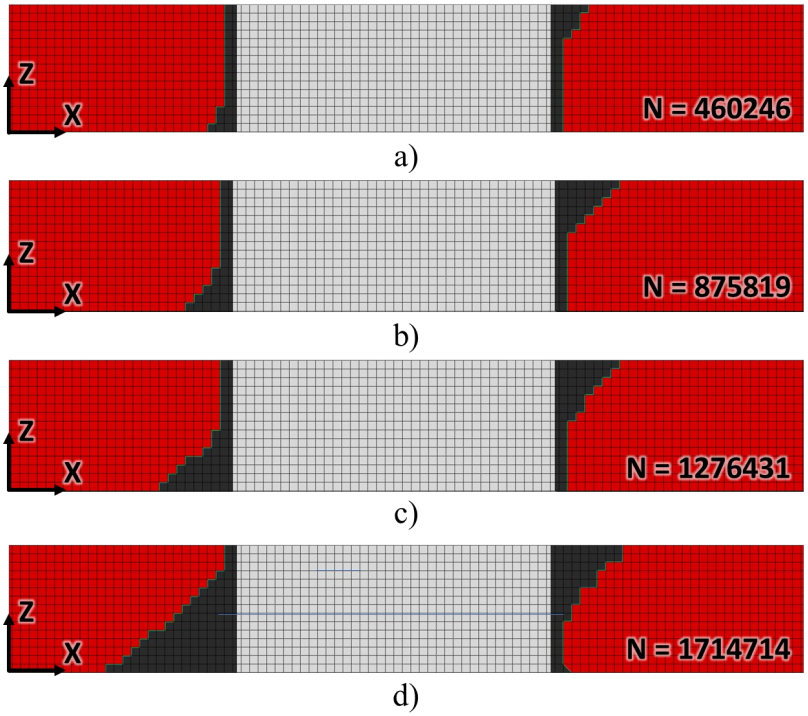

Figure 17: Delamination in the composite panel after: a) 460246, b) 875819, c) 1276431 and d) 1714714 cycles.

6 Conclusions and future works

A new VCCT-based algorithm, called Sequential-Static Fatigue, to model the delamination growth under fatigue loadings was proposed in this work. The main purpose of its development was to reduce the computational time of the DC algorithm present in Abaqus, considered as benchmark. The use of the SSF algorithm led to a drastic time reduction of about three orders of magnitude: for the same simulations, the DC algorithm required several days of computational time, while the SSF algorithm only required a few minutes. The SSF algorithm proved also to be more accurate than its counterpart, since it is based on

general static simulations, which are more accurate than the cyclic ones adopted by the DC algorithm. Overall, the advantages of the SSF algorithm over the DC one are:

- a significantly increase in the computational speed between two and three order of magnitudes;
- the versatility of implementing any Paris-like propagation law for different cases;
- the ability to consider a different set of Paris parameters for each node, based on the local values of $\mathcal{G}_{\mathrm{I}}$, $\mathcal{G}_{\mathrm{II}}$, and $\mathcal{G}_{\mathrm{III}}$.

The SSF algorithm was validated against a benchmark test case, proving fair predictions capabilities. The main inaccuracies of this validation were due to the mesh sensitivity of the VCCT in the static simulations, and not to inherent defects of the SSF. The SSF algorithm performed well also in the case of a real-life structure, proving its potential of application in industrially relevant scenarios.

While the purpose of this work was to present the SSF algorithm and prove its basic capabilities, the algorithm has several interesting potentialities worth investigating. This is mainly due to the versatility of the algorithm, that allows the implementation of any Paris law suitable for different applications. Currently, the following aspects are under investigation:

- the SSF capabilities of predicting the effect of the load ratio R on the delamination growth;
- the possibility of adopting the SSF algorithm for debonding of adhesively bonded joints.

Overall, this work set the basis for a new efficient VCCT-based modelling approach for delamination of composite materials and structures under fatigue loadings.

7 Acknowledgments

The authors wish to thank J. A. Pascoe (TU Delft) for the useful discussions and suggestions on the topic.

8 References

[1] Harris B. Fatigue in Composites - Science and Technology of the Fatigue Response of Fibre-Reinforced Plastics. Cambridge, England, England: Woodhead Publishing Limited; 2003.
[2] Jokinen J, Kanerva M. Simulation of Delamination Growth at CFRP-Tungsten Aerospace Laminates Using VCCT and CZM Modelling Techniques. Appl Compos Mater 2019;26:709–21.
[3] Turon A, Costa J, Camanho PP, Dávila CG. Simulation of delamination in composites under

high-cycle fatigue. Compos Part A Appl Sci Manuf 2007;38:2270–82.
[4] Bak BLV, Turon A, Lindgaard E, Lund E. A benchmark study of simulation methods for high-cycle fatigue-driven delamination based on cohesive zone models. Compos Struct 2017;164:198–206.
[5] Pirondi A, Giuliese G, Moroni F, Bernasconi A, Jamil A. Comparative study of cohesive zone and virtual crack closure techniques for three-dimensional fatigue debonding. J Adhes 2014;90:457–81.
[6] Banea MD, da Silva LFM. Adhesively bonded joints in composite materials: An overview. Proc Inst Mech Eng Part L J Mater Des Appl 2009;223:1–18.
[7] Budhe S, Banea MD, de Barros S, da Silva LFM. An updated review of adhesively bonded joints in composite materials. Int J Adhes Adhes 2017;72:30–42.
[8] Harper PW, Hallett SR. A fatigue degradation law for cohesive interface elements – Development and application to composite materials. Int J Fatigue 2010;32:1774–87.
[9] Rybicki EF, Kanninen MF. A finite element calculation of stress intensity factors by a modified crack closure integral. Eng Fract Mech 1977;9:931–8.
[10] Krueger R. Virtual crack closure technique: History, approach, and applications. Appl Mech Rev 2004;57:109–43.
[11] Sadeghi MZ, Gabener A, Zimmermann J, Saravana K, Weiland J, Reisgen U, et al. Failure load prediction of adhesively bonded single lap joints by using various FEM techniques. Int J Adhes Adhes 2020;97:102493.
[12] Jokinen J, Wallin M, Saarela O. Applicability of VCCT in mode I loading of yielding adhesively bonded joints—a case study. Int J Adhes Adhes 2015;62:85–91.
[13] Eder MA, Bitsche RD. Fracture analysis of adhesive joints in wind turbine blades. Wind Energy 2015;18:1007–22.
[14] Shokrieh MM, Heidari-Rarani M, Rahimi S. Influence of curved delamination front on toughness of multidirectional DCB specimens. Compos Struct 2012;94:1359–65.
[15] Shokrieh MM, Rajabpour-Shirazi H, Heidari-Rarani M, Haghpanahi M. Simulation of mode I delamination propagation in multidirectional composites with R-curve effects using VCCT method. Comput Mater Sci 2012;65:66–73.
[16] Tawk I, Navarro P, Ferrero J-F, Barrau J-J, Abdullah E. Composite delamination modelling using a multi-layered solid element. Compos Sci Technol 2010;70:207–14.
[17] Pascoe JA, Alderliesten RC, Benedictus R. Methods for the prediction of fatigue delamination growth in composites and adhesive bonds – A critical review. Eng Fract Mech 2013;112–113:72–96.
[18] Freed Y, Zobeiry N, Salviato M. Development of aviation industry-oriented methodology for failure predictions of brittle bonded joints using probabilistic machine learning. Compos Struct 2022:115979.
[19] Jamil A. Assessment of the VCCT technique for modelling fatigue crack growth in adhesively bonded composite materials. Politecnico di Milano, Doctoral Thesis, 2014.
[20] Krueger R. An Approach to Assess Delamination Propagation Simulation Capabilities in Commercial Finite Element Codes. Tech. Rep. NASA/TM-2008-215123: 2008.
[21] Siemens Digital Industries Software. Simcenter Samcef 2021.
[22] Dassault Systemès. Abaqus 2021 manual. n.d.
[23] Giuliese G, Pirondi A, Moroni F, Bernasconi A, Jamil A, Nikbakh A. Fatigue delamination: a comparison between Virtual Crack Closure and Cohesive Zone siulation techniques. 19th Int. Conf. Compos. Mater., Montréal, Canada, Canada: 2013.
[24] Raimondo A, Doesburg SA, Bisagni C. Numerical study of quasi-static and fatigue delamination growth in a post-buckled composite stiffened panel. Compos Part B Eng 2020;182:107589.
[25] Vescovini R, Dávila CG, Bisagni C. Failure analysis of composite multi-stringer panels using simplified models. Compos Part B Eng 2013;45:939–51.
[26] Dávila CG, Bisagni C. Fatigue life and damage tolerance of postbuckled composite stiffened structures with initial delamination. Compos Struct 2017;161:73–84.
[27] Benzeggagh ML, Kenane M. Measurement of mixed-mode delamination fracture toughness of unidirectional glass/epoxy composites with mixed-mode bending apparatus. Compos Sci Technol 1996;56:439–49.
[28] Hosseini-Toudeshky H, Sheibanian F, Ovesy HR, Goodarzi MS. Prediction of interlaminar fatigue damages in adhesively bonded joints using mixed-mode strain based cohesive zone modeling. Theor Appl Fract Mech 2020;106:102480.

[29] Blanco N, Gamstedt EK, Asp LE, Costa J. Mixed-mode delamination growth in carbon–fibre composite laminates under cyclic loading. Int J Solids Struct 2004;41:4219–35.
[30] Pirondi A, Moroni F. Simulation of mixed-mode I/II fatigue crack propagation in adhesive joints with a modified cohesive zone model. J Adhes Sci Technol 2011;25:2483–99.
[31] Bak BL V., Turon A, Lindgaard E, Lund E. A simulation method for high-cycle fatigue-driven delamination using a cohesive zone model. Int J Numer Methods Eng 2016;106:163–91.
[32] Rocha A, Akhavan-Safar A, Carbas R, Marques E, Goyal R, El-Zein M, et al. Paris law relations for an epoxy-based adhesive. Proc Inst Mech Eng Part L J Mater Des Appl 2020;234:291–9.
[33] Python 3.6 2021. https://www.python.org/.
[34] Asp L, Sjögren A, Greenhalgh E. Delamination Growth and Thresholds in a Carbon/Epoxy Composite Under Fatigue Loading. J Compos Technol Res 2001;23:55.
[35] Williams JG. On the calculation of energy release rates for cracked laminates. Int J Fract 1988;36:101–19.
[36] Robinson P, Galvanetto U, Tumino D, Bellucci G, Violeau D. Numerical simulation of fatigue-driven delamination using interface elements. Int J Numer Methods Eng 2005;63:1824–48.
[37] Carreras L, Renart J, Turon A, Costa J, Bak BLV, Lindgaard E, et al. A benchmark test for validating 3D simulation methods for delamination growth under quasi-static and fatigue loading. Compos Struct 2019;210:932–41.
[38] Carreras L, Renart J, Turon A, Costa J, Bak BLV, Lindgaard E, et al. A benchmark test for validating 3D simulation methods for delamination growth under quasi-static and fatigue loading. Compos Struct 2019;210:932–41.
[39] Raimondo A, Dávila CG, Bisagni C. Cohesive analysis of a 3D benchmark for delamination growth under quasi-static and fatigue loading conditions. Fatigue Fract Eng Mater Struct 2022;45:1942–52.
[40] Bisagni C, Dávila CG. Experimental investigation of the postbuckling response and collapse of a single-stringer specimen. Compos Struct 2014;108:493–503.
[41] Daricik F. Mesh size sensitivity analysis for interlaminar fracture of the fiber-reinforced laminated composites. J Eng Fiber Fabr 2019;14:155892501988346.