Switching max-plus models for legged locomotion

G.A.D. Lopes, R. Babuška, B. De Schutter, and A.J.J. van den Boom

If you want to cite this report, please use the following reference instead:

Delft University of Technology
Delft Center for Systems and Control
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

*This report can also be downloaded via https://pub.deschutter.info/abs/09_045.html
Switching Max-Plus Models for Legged Locomotion

G.A.D. Lopes, R. Babuška, B. De Schutter and A.J.J. van den Boom

Abstract—We present a new class of gait generation and control algorithms based on the Switching Max-Plus modeling framework that allows for the synchronization of multiple legs of walking robots. Transitions between stance and swing phases of each leg are modeled as discrete events on a system described by max-plus-linear state equations. Different gaits and gait parameters can be interleaved by using different system matrices. Switching in max-plus-linear systems offers a powerful collection of modeling, analysis, and control tools that, in particular, allow for safe transitions between different locomotion gaits that may involve breaking/enforcing synchronization or changing the order of leg lift off events. Experimental validation of the proposed algorithms is presented by the implementation of various horse gaits on a simple quadruped robot.

1. INTRODUCTION

The field of discrete event systems (DES) [1] finds its core of applications in scheduling problems from engineering such as manufacturing, communications, traffic, computer systems, etc. Typically, the set of equations that describes the dynamics of such systems are nonlinear in traditional algebra. However, there is a subclass of timed DES (classes of discrete event systems where there exists an underlying time structure) that can be framed in sets of linear equations for a different type of algebra. These are called max-plus-linear discrete event systems (MPL-DES) defined in the max-plus algebra [2]–[4]. Systems that enforce synchronization have no concurrency can be modeled in this framework. Systems that can be modeled as MPL-DES inherit a large set of analysis and control synthesis tools thanks to many parallels between the max-plus-linear systems theory and the traditional linear systems theory.

Legged locomotion systems can be elegantly modeled by limit cycles on cross products of circles, due to their intrinsic periodic nature (see “networks of phase oscillators” and “central pattern generators” (CPG) in Holmes et al. [5] or the earlier works of Grillner [6] and Cohen et al. [7]). One can represent the position of each leg at any given instant by mapping it into a phase in a circle. Synchronization can be achieved by enforcing phase differences on the circles. This modeling framework is very natural for both land and water locomotion (or even flight), but entails the construction of an “anchoring” map [8]. The biology community has extensively explored these concepts to classify the different gaits of horses [9], insects [10], and other animals [11]. The robotics community has harvested this knowledge to develop different techniques of generating motion in legged robots. Klavins et al. [12] show how to systematically generate vector fields that reach piecewise constant velocity limit cycles, Erden et al. [13] use reinforcement learning tools on a hexapod robot, and Zhao et al. [14] use CPG models for the control of a biomimetic fish. For an extensive literature review on legged locomotion see [5].

An interesting analogy can be made between enforcing phase differences in continuous cycles and synchronization in timed discrete event processes. In a typical (continuous) walking motion of a biped robot, the left leg should only lift off the ground after the right leg has touched down, to make sure the robot does not fall from lack of support1. This synchronization requirement can be modeled by the evolution of a discrete event system by abstracting each limit cycle in the circle into two sequential events in a closed circuit: lift off and touchdown. Each leg is then modeled by an event cycle and synchronization between legs is enforced by connections between each leg cycle. Synchronization of multiple legs is especially important for climbing robots [15], [16] where lack of support can result in catastrophic consequences. In this paper we show how the max-plus framework can be naturally utilized to systematically implement motion gaits for legged locomotion with guaranteed synchronization by design.

Some work has been done on low-level walking gait generation from a DES point of view such as [17], [18], and using Petri nets in [19]. In these implementations the focus is put on generating each individual gait. In this paper we take advantage of the properties of switching max-plus-linear models [20] to not only generate locomotion gaits but more importantly to deal with the transitions between different gaits and recovering from large perturbations. In a switching max-plus-linear system one can interchange different modes of operation. In each mode the discrete event system is described by a max-plus-linear state space model with different system matrices. In this application a mode corresponds to a specific gait.

Gait transition has been studied in biology from an energetic point of view [21] and in the robotics field was approached informally by Raibert et al. [22]. In this paper, we show that safe gait transitions arise naturally from the max-plus framework.

In Section II we introduce the tools behind max-plus systems, and we show in Section III how these can be used...
II. MAX-PLUS ALGEBRA

We start by revising the structure of the max-plus algebra. Let $\varepsilon \equiv -\infty$, $e \equiv 0$, and $R_{\text{max}} = R \cup \{\varepsilon\}$. Define the operations $\oplus, \otimes : R_{\text{max}} \times R_{\text{max}} \rightarrow R_{\text{max}}$ by:

\[
\begin{align*}
x \oplus y &\equiv \max(x, y) \\
x \otimes y &\equiv x + y
\end{align*}
\]

**Definition 1:** The set $R_{\text{max}}$ with the operations $\oplus$ and $\otimes$ is called the max-plus algebra, denoted by $R_{\text{max}} = (R_{\text{max}}, \oplus, \otimes, e, \varepsilon)$.

**Theorem 1 ([3]):** The max-plus algebra $R_{\text{max}}$ has the algebraic structure of a commutative idempotent semiring.

The max-plus algebra can be interpreted as the traditional linear algebra with the operations ‘$+$’ and ‘$\times$’ replaced by the operators ‘$\max$’ and ‘$+$’, respectively, with the supplemental difference that the additive inverse does not exist, thus resulting in a semiring. Matrices can be defined by taking Cartesian products of $R_{\text{max}}$ and denoting $A \in R_{\text{max}}^{n \times m}$. Let $a_{ij} = [A]_{ij}$ be the $i,j$ element of $A$. For $A, B \in R_{\text{max}}^{n \times m}$ and $C \in R_{\text{max}}^{m \times p}$ define the matrix sum $\oplus$ and matrix product $\otimes$ operations by:

\[
\begin{align*}
[A \oplus B]_{ij} &= a_{ij} \oplus b_{ij} \equiv \max(a_{ij}, b_{ij}) \\
[A \otimes C]_{ij} &= \bigoplus_{k=1}^{m} a_{ik} \otimes c_{kj} \equiv \max_{k=1, \ldots, m} (a_{ik} + c_{kj})
\end{align*}
\]

Switching max-plus-linear systems are described by a state space model

\[
x(k + 1) = A_{(m(k))} \otimes x(k),
\]

where the state $x(k)$ typically contains the time instants at which the internal events occur for the $k$-th time, and $A_{(m(k))}$ is the system matrix for mode $m(k)$. In this walking robot application the system matrix $A_{(m(k))}$ is in a set of locomotion gaits indexed by $m(k)$.

III. QUADRUPED GAITS

The first step towards modeling locomotion gaits is to define the state variables for the transition events. Let $l_i(k)$ be the time instant leg $i$ lifts off the ground and $l_i(k)$ be the time instant it touches the ground, both for the $k$-th iteration. For a traditional alternating swing/stance gait one can impose that the time instant when the leg touches the ground must equal the time instant it lifted off the ground for the last time plus the time it stays in flight (denoted $\tau_f$):

\[
l_i(k) = l_i(k) + \tau_f
\]

Analogously, we get a similar relation for the lift off time:

\[
l_i(k) = l_i(k - 1) + \tau_g
\]

where $\tau_g$ is the stance time and $l_i$ uses the previous iteration such that equations (2) and (3) can be used iteratively. Suppose now that one aims to synchronize leg $i$ with leg $j$ in such a way that leg $i$ can only lift off after leg $j$ has touched the ground. One can then write the relation:

\[
l_i(k) = \max(l_i(k - 1) + \tau_g, l_j(k - 1) + \tau_\Delta)
\]

Equation (4) enforces simultaneously that both the leg $i$ stays at least $\tau_g$ seconds in stance and will only lift off at least $\tau_\Delta$ seconds after leg $j$ has touched down. When both conditions are satisfied, lift off takes place. Following this reasoning, one can efficiently represent motion gaits in terms of synchronization of timed events. Figure 2 illustrates three different gaits for quadrupeds: pacing, trotting, and walking (according to the leg numbering in Figure 1), and Table I compiles the state variables and the associated parameters used throughout this paper. Pacing, commonly used by camels and dromedaries, is a lateral two-beat gait, i.e., front and back legs are synchronized and opposite in phase to front and back right legs. In trotting, opposite legs are synchronized and we assume that for the walking gait at least three legs are on the ground at all times.

The traditional pacing gait with no aerial phase is illustrated in Figure 2.a. The arrows represent the relationship between events that must occur for leg lift off to happen.

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>STATE VARIABLES AND GAIT PARAMETERS.</strong></td>
</tr>
<tr>
<td>$x(k)$</td>
</tr>
<tr>
<td>$l_i(k)$</td>
</tr>
<tr>
<td>$l_i(k)$</td>
</tr>
<tr>
<td>$i$</td>
</tr>
<tr>
<td>$\theta_i$</td>
</tr>
<tr>
<td>$\theta_i$</td>
</tr>
<tr>
<td>$\tau$</td>
</tr>
<tr>
<td>$\tau_f$</td>
</tr>
<tr>
<td>$\tau_g$</td>
</tr>
<tr>
<td>$\tau_1, \tau_p, \tau_w$</td>
</tr>
<tr>
<td>$k_j$</td>
</tr>
</tbody>
</table>

Fig. 1. Perspective and top view of a quadruped robot developed at DCSC, TU Delft, inspired by the hexapod robot RHex [23]. The roman numbers on the right picture represent the leg ordering.
Following the notation in Figure 2.a) one obtains the timed event equations for the pacing gait:

\[
\begin{align*}
  t_1(k+1) &= l_1(k+1) + \tau_f \\
  l_1(k+1) &= \max(t_1(k) + \tau_g, l_2(k) + \tau_p, t_4(k) + \tau_p) \\
  t_2(k+1) &= l_2(k+1) + \tau_f \\
  l_2(k+1) &= \max(t_2(k) + \tau_g, l_1(k+1) + \tau_p, t_3(k+1) + \tau_p) \\
  t_3(k+1) &= l_3(k+1) + \tau_f \\
  l_3(k+1) &= \max(t_3(k) + \tau_g, t_2(k) + \tau_p, t_4(k) + \tau_p) \\
  t_4(k+1) &= l_4(k+1) + \tau_f \\
  l_4(k+1) &= \max(t_4(k) + \tau_g, l_1(k+1) + \tau_p, t_3(k+1) + \tau_p).
\end{align*}
\]

The previous set of equations can be transformed into the structure of equation (1) by recursive substitution. For example, the update equation for \( t_1 \) becomes:

\[
\begin{align*}
  t_1(k+1) &= l_1(k+1) + \tau_f \\
  &= \max(t_1(k) + \tau_g, t_2(k) + \tau_p, t_4(k) + \tau_p) + \tau_f \\
  &= \max(t_1(k) + \tau_g + \tau_f, t_2(k) + \tau_p + \tau_f, t_4(k) + \tau_p + \tau_f).
\end{align*}
\]

Following the same recursive process for the other variables, and defining the state variables \( x(k) \in \mathbb{R}_{\text{max}}^n \) by

\[
  x(k) = [t_1(k), l_1(k), \ldots, t_4(k), l_4(k)]^T,
\]

one can find the max-plus-linear system matrix for the pacing gait, that we denote \( A_p \). For gait symmetry we assume that

\[
\tau_g > \tau_f, \quad \text{and} \quad \tau_p = \frac{\tau_g - \tau_f}{2}.
\]

The extra parameters \( \tau_{fg} = \tau_f + \tau_g \) and \( \tau_{pf} = \tau_p + \tau_f \) are introduced for compactness.

Solving system (1) for the pacing gait \( A_p \) with all time events initialized to zero, and with parameters \( \tau_g = 3s \) and \( \tau_f = 1s \), results in the following leg scheduling:

<table>
<thead>
<tr>
<th>( k )</th>
<th>( t_1 )</th>
<th>( l_1 )</th>
<th>( l_2 )</th>
<th>( l_3 )</th>
<th>( l_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>11</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

After a single iteration all the synchronization conditions are met. See Appendix I for the system matrices of trotting and walking gaits.

IV. Timed Event Graphs

The theory of Petri nets [1] provides an intuitive tool for representing DES graphically: the Petri net graphs. It is straightforward to generate walking gaits by evolving a Petri net in time if: 1) each leg is modeled as a circuit with lift off and touchdown transitions 2) synchronizations between legs are implemented by firing conditions on the lift off transition. Moreover, by carefully designing the Petri net one can take advantage of the synthesis tools of max-plus algebras. A subclass of timed Petri nets called event graphs are formally equivalent to max-plus linear systems [3]. This allows for the intuitive design of locomotion gaits by event graphs with a direct translation to max-plus linear systems and vice versa.

Definition 2 ([4]): A timed Petri net \( \mathcal{G} \) is characterized by a set of places \( \mathcal{P} \), a set of transitions \( \mathcal{Q} \), a set of arcs \( \mathcal{D} \) from transitions to places and vice versa, an initial marking \( M_0 \), and a holding time vector \( T \). If each place has exactly one upstream and one downstream transition, then the timed Petri net is called a timed event graph.

Figure 3 illustrates the timed event graph for the pacing gait described in Figure 2. The construction of such a graph goes as follows:

1) For each leg define a circuit with the two events: touchdown \( T_i \) and lift off \( L_i \). Between the events add the
places $G_i$ for the time the leg $i$ stays in the ground and $F_i$ for the time it is in flight. In Figure 3 this is represented by the four numbered thicker circuits.

2) For each required synchronization add a place between transitions. It is important to make sure that each place has a unique upstream arc and a unique downstream arc. For example, the place labeled $S_{14}$ enforces that the lift off transition of leg ‘iv’ only fires after leg ‘i’ has touched the ground.

3) Initialize the marking such that all the ground places have a token (the robot starts by having all the legs on the ground). Moreover, tokens are added to synchronization places such that all (closed) circuits are alive and the initial marking is feasible. For the case of the pacing gait, four tokens are added such that the lift off transitions of legs ‘ii’ and ‘iv’ are ready to fire in the initial marking.

Following the steps presented above, it is possible to reproduce many locomotion gaits. Once the timed event graph is constructed it is straightforward to find its associated max-plus-linear system [4]:

1) Each transition $\Psi_i$, in the timed event graph is assigned to a state variable $\psi_i$ in the max-plus algebra.

2) For each incoming arc to a transition $\Psi_i$, make a list of
   - the event $\Psi_j$ that precedes the event $\Psi_i$ for that arc and add $j$ to a set $S$
   - the time constant $\psi_j$ of the place of the arc’s origin
   - the number of tokens $\kappa_j$ of the place of the arc’s origin

Then write the expressions:

$$\psi_i(k+1) = \max_{j \in S} (\psi_j(k+1 - \kappa_j) + \psi_j)$$

(5)

For example, consider the transition $T_1$ in Figure 3. It has a single incoming arc from the transition $L_1$ with time constant $\tau_f$ and zero tokens. Thus, equation (5) for this transition is

$$t_1(k+1) = l_1(k+1) + \tau_f$$

For transition $L_1$ we have three incoming arcs from $T_1$, $T_2$, and $T_3$, with respective time constants $(\tau_g, \tau_p, \tau_p)$ and one token per place. The transition equation for $l_2$ is then:

$$l_1(k+1) = \max (l_1(k) + \tau_g, t_2(k) + \tau_p, t_4(k) + \tau_p)$$

Following this procedure one obtains exactly the transition equations for the pacing gait derived in Section III. Note that if the four synchronization tokens added in the initial markup to the places $\{S_{21}, S_{23}, S_{41}, S_{41}\}$ were instead placed in $\{S_{12}, S_{14}, S_{32}, S_{34}\}$ the resulting equations would be a change of coordinates away from the pacing equations in Section III. The same would be true for any resulting token configuration after any feasible firing of the timed event graph in Figure 3. The remaining combinations of tokens either block the Petri net or break the required synchronization resulting in an incorrect time evolution for the gait.

---

**Fig. 3.** Timed event graph for a pacing gait. Leg lift off events are represented by $L_1$ and touchdown events by $T_i$. Each place has a label name on top and a holding time on the bottom. Ground places are represented by $G_i$, flight places by $F_i$, and leg synchronization places by $S_i$. The thick numbered loops represent the discrete event periodic cycles for each leg.

---

### V. Control Structure

The max-plus-linear system for locomotion gaits derived in Section III returns a state vector of the time instants when leg touchdown and lift off events must occur in the future. These must be translated into continuous-time trajectories for the control of each motor. Moreover, the discrete event state variables must be updated by the continuous-time state variables (angles of the legs) to measure the true time the events occurred and recompute the future event timings. Figure 4 presents the block diagram of the hybrid control structure we propose in this paper. The supervisory control block generates max-plus-linear system matrices. The max-plus gait scheduler block implements system (1). The continuous time scheduler generates a continuous time reference trajectory for the events generated by the max-plus scheduler. Finally, the reference trajectory tracker implements local PD controllers at each motor to track the reference trajectory.

#### A. Continuous-time trajectory generator

The timed event equations derived in Section III must be mapped into the continuous-time domain of a legged robot. This can be accomplished by defining a reference trajectory generating function

$$\theta_{ref} : \mathbb{R}^+ \times (\mathbb{R}_{\max}^{2n})^p \rightarrow (S^1)^n$$

that takes as inputs time $\tau \in \mathbb{R}^+$ plus a collection of $p$ discrete events $x(k) \in \mathbb{R}_{\max}^{2n}$ and outputs a piecewise linear function.

---

*In this paper the supervisory control box is replaced by a human selecting different locomotion gaits through a graphical user interface.*
trajectory for each of the leg angles, with $\theta_1 < \theta_2$:

$$\theta_{\text{ref},i}(\tau) = \begin{cases} 
\theta_l (t_i(k_{2i-1}) - \tau) + (\theta_1 + 2\pi) (\tau - t_i(k_{2i-1})) & \text{if } \tau \in [t_i(k_{2i-1}), l_i(k_{2i-1})] \\
\theta_l (l_i(k_{2i-1}) + 1) - \theta_l (t_i(k_{2i-1})) + \theta_1 (\tau - t_i(k_{2i-1})) & \text{if } \tau \in [l_i(k_{2i-1}), l_i(k_{2i-1} + 1)] 
\end{cases}$$

The function $\theta_{\text{ref}}$ takes a $p$-collection of events since there is no necessity for the intervals $[l_i(k_{2i-1}), l_i(k_{2i-1} + 1)]$ to be overlapping for all legs. The event indices $\{k_j\} \in \mathbb{N}^8$ are chosen for each leg such that the time $\tau$ lies in the proper interval.

**B. Feedback control**

Feedback control is implemented in both the reference trajectory tracker and max-plus gait scheduler blocks represented in the diagram of Figure 4. In the first, a reference trajectory feedback control loop is implemented by a simple PD controller. In the second, events of touchdown and lift off are measured by observing when the legs cross specific angles during their motion. The time the true event occurred is updated into the max-plus gait scheduler which then recomputes the subsequent event transition times.

**C. Switching gaits**

As previously described, each gait is encoded in the system matrices $A_{(m(k))}$ of equation (1). In this paper we have

$$A_{(m(k))} \in A = \{A_p, A_t, A_w\},$$

Where the indices represent pacing, trotting and walking.

The max-plus algebraic representation guarantees that each transition is safe by construction. Note that the system matrices are parameterized by the time constants that encode leg flight time, ground time, and other leg offsets:

$$A_{(m(k))} = A_{(m(k))}(\tau_f, \tau_g, \ldots).$$

Thus, beyond changing gaits, one can also safely change the parameters of $A$ at each iteration $k$, resulting in different locomotion speeds or different leg offset synchronizations.

**VI. EXPERIMENTAL RESULTS**

The tools presented in this paper were implemented in a small quadruped robot developed at TU Delft, illustrated in Figure 1. The morphology is inspired by the hexapod RHx robot [23] albeit with a different number of legs. The algorithms are implemented in Matlab, running on a small form factor laptop PC that stands on the robot. A Matlab GUI is used to switch the locomotion gaits and their associated parameters.

**VII. CONCLUSIONS AND FUTURE WORK**

We have presented a new modeling tool for locomotion of multi-legged robots based on the max-plus algebra. Gait design and implementation can be efficiently accomplished by connected circuits of discrete events. Leg synchronization is guaranteed by design of the max-plus framework. We show that gait switching is efficiently implemented by simply switching system matrices in the evolution of a maxplus-linear system. Experimental results in locomotion are presented for validation of the proposed framework.

Although the present implementation uses the max-plus framework as a modeling technique that achieves a compact notation and efficient implementation, we aim in future work to fully explore the analysis and synthesis tools of the max-plus algebras to improve on the switching behaviors of legged locomotion beyond safety. As an example, by looking at the eigenstructure of the system matrices of a max-plus-linear system it is possible extract very useful data that can inform a supervisory control of the “best”3 gait to switch to, at any given instant. We are currently developing such a supervisor controller.

**APPENDIX I**

**SYSTEM MATRICES FOR TROTTING AND WALKING GAITS**

**A. Trotting gait**

The trotting gait, illustrated in Figure 2.b) is analogous to the pacing gait. Again we choose $\tau_l = (\tau_g - \tau_f)/2$ and $\tau_i$ as

3In terms of switching speed, i.e. pick the gait that introduces the least amount of delay into the event transitions.
\( \tau_f = \tau + \tau_f \). The system matrix \( A_t \) for this gait is:

\[
A_t = \begin{bmatrix}
\tau_{fg} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{fg} & \epsilon \\
\tau_g & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_g & \epsilon \\
\tau_{fg} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{fg} & \epsilon \\
\tau_g & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_g & \epsilon \\
\tau_{fg} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{fg} & \epsilon \\
\tau_g & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_{f+\tau_g} & \epsilon & \tau_g & \epsilon
\end{bmatrix}
\]

**B. Walking gait**

For the walking gait, illustrated in Figure 2.c) we assume that at least three legs must be on the ground at all times. Choosing the timing parameter

\[
\tau_g > 3\tau_f, \quad \tau_w = \frac{\tau_g - 3\tau_f}{4}
\]

results in a symmetrical gait. Defining \( \tau_{wf} = \tau_w + \tau_f \), the system matrix \( A_w \) for the walking gait is:

\[
A_w = \begin{bmatrix}
4\tau_{wf} & 6\tau_{wf} & 5\tau_{wf} & 3\tau_{wf} & \epsilon \\
3\tau_{wf} + \tau_w & 5\tau_{wf} + \tau_w & 4\tau_{wf} + \tau_w & 3\tau_{wf} + \tau_w & \epsilon \\
2\tau_{wf} + \tau_w & 4\tau_{wf} + \tau_w & 3\tau_{wf} + \tau_w & 2\tau_{wf} + \tau_w & \epsilon \\
3\tau_{wf} + \tau_w & 5\tau_{wf} + \tau_w & 4\tau_{wf} + \tau_w & 3\tau_{wf} + \tau_w & \epsilon \\
2\tau_{wf} + \tau_w & 4\tau_{wf} + \tau_w & 3\tau_{wf} + \tau_w & 2\tau_{wf} + \tau_w & \epsilon \\
\tau_{wf} & 3\tau_{wf} & 2\tau_{wf} & \tau_{wf} & \epsilon \\
\tau_w & 2\tau_{wf} + \tau_w & \tau_{wf} + \tau_w & \tau_{wf} & \epsilon
\end{bmatrix}
\]

**REFERENCES**


