Alma Mater Studiorum – Universitá di Bologna

DOCTORAL THESIS IN

MECHANICAL ENGINEERING

Ciclo XXV

ING-IND/14

The Dynamics

of

Passive Torsional Fatigue Test Rigs

Innovative Applications of Universal Joints

Candidate	Supervisor
Carlo Peressini	Prof. Pier Gabriele Molari

Co-supervisors

A/Prof. Dianne Hesterman

Dr. Andrew Guzzomi

March, 2013

To my family

Abstract

The dynamics of a passive back-to-back test rig have been characterised, leading to a multi-coordinate approach for the analysis of arbitrary test configurations. Universal joints have been introduced into a typical pre-loaded back-to-back system in order to produce an oscillating torsional moment in a test specimen. Two different arrangements have been investigated using a frequency-based sub-structuring approach: the receptance method. A numerical model has been developed in accordance with this theory, allowing interconnection of systems with two-coordinates and closed multi-loop schemes. The model calculates the receptance functions and modal and deflected shapes of a general system. Closed form expressions of the following individual elements have been developed: a servomotor, damped continuous shaft and a universal joint. Numerical results for specific cases have been compared with published data in literature and experimental measurements undertaken in the present work. Due to the complexity of the universal joint and its oscillating dynamic effects, a more detailed analysis of this component has been developed. Two models have been presented. The first represents the joint as two inertias connected by a massless cross-piece. The second, derived by the dynamic analysis of a spherical four-link mechanism, considers the contribution of the floating element and its gyroscopic effects. An investigation into non-linear behaviour has led to a time domain model that utilises the Runge-Kutta fourth order method for resolution of the dynamic equations. It has been demonstrated that the torsional receptances of a universal joint, derived using the simple model, result in representation of the joint as an equivalent variable inertia. In order to verify the model, a test rig has been built and experimental validation undertaken. The variable inertia of a universal joint has lead to a novel application of the component as a passive device for the balancing of inertia variations in slider-crank mechanisms.

Contents

List of Figures

List of Tables

Symbols & Units

CHAPTER 2

Δ	frequency expression
α, β, χ	receptances component
$|\boldsymbol{\alpha}|$	receptance matrix
F_q	generalised force at coordinate q
$\boldsymbol{F}$	generalised force vector
$\boldsymbol{u}$	generalised displacement vector
u_p	generalised displacement at coordinate p

CHAPTER THREE - SECTION 3.2

Δl	variation of length	[m]
$\ddot{\Theta}$	angular acceleration amplitude	[rad/s^2]
η	axial hysteretic damping ratio	
μ	torsional hysteretic damping ratio	
ω	angular frequency	[rad/s]
ρ	density	[kg/m^3]
θ	angular displacement	[rad]
ε	deformation	

ξ	damping ratio	
A	section area	[m^2]
E	Young's modulus	[N/m^2]
F	external axial force	[N]
G	shear modulus	[N/m^2]
H	hysteretic damping force	[N]
L	bar length	[m]
N	internal axial force	[N]
$\dot{s}$	velocity	[m/s]
F	force amplitude	[N]
U	displacement amplitude	[m]
k	stiffness	[N/m]
l	length	[m]
s	displacement	[m]
u	axial displacement	[m]

CHAPTER 3 - SECTION 3.3.1

α_{pq}	torsional receptance	[Nm/rad]
α	angular displacement of cross piece input axis	[rad]
β	angular displacement of cross piece output axis	[rad]
$\ddot{\phi}$	angular acceleration	[rad/s^2]
$\dot{\phi}$	angular velocity	[rad/s]
ω_{i}	discrete angular frequency	[rad/s]
ω	angular frequency	[rad/s]

ϕ_a	input angular displacement	[rad]
ϕ_b	output angular displacement	[rad]
τ	transmission ratio	
θ	misalignment angle	[rad]
I'_{ra}	derivation of I_{ra} with respect to ϕ_a	[kg m^2/rad]
I'_{rb}	derivation of I_{rb} with respect to ϕ_b	[kg m^2/rad]
I_a	inertia about axis a	[kg m^2]
I_b	inertia about axis b	[kg m^2]
I_{ra}	equivalent inertia resolved to axis a	[kg m^2]
I_{rb}	equivalent inertia resolved to axis b	[kg m^2]
M_{xa}	component of moment M_x in a direction	[Nm]
M_{xb}	component of moment M_x in b direction	[Nm]
M_x	moment acting on the crosspiece	[Nm]
T	applied external torque	[Nm]
T_a	applied external torque at node a	[Nm]
T_b	applied external torque at node b	[Nm]
f$_i$	discrete frequency	[Hz]
k$_m$	servomotor torsional stiffness	[Nm/rad]
k	torsional stiffness	[Nm/rad]
j^2	$= -1$	
p,q	subscripts referring to coordinates	

CHAPTER 4

δ	universal joint misalignment angle	[rad]

θ_{PL}	imposed angular displacement	[rad]
ξ	percentage of critical damping for spur gear pairs	
I	inertia	[kg m^2]
M_{PL}	locked moment within the rig	[Nm]
$M_{\{\}}$	static moment	[Nm]
$R_{\{\}}$	gear radius	[m]
c	viscous damping	[Nm/rad/s]
i	gear ratio	

1 | Introduction

1.1 Background and motivation

With the advent of automatic machines at the end of XVIII century in the textile and metallurgic sectors, mechanical component failures due to fatigue loading cycles under the 'breaking weight' [6] were documented. In 1837, Albert published fatigue-test results on conveyor chain damage in Clausthal [7]. To date, this work appears to be the first publication in the field. It is furthermore of interest to note that 'he tested actual components, not just the material' [8]. Since then, more in-depth investigations have been conducted into the fatigue of metals in order to explain the physics of the phenomenon and to give engineers more advanced design tools. Nonetheless, fatigue failures continue to occur during *in situ* operation, incurring significant costs to industry and presenting safety risks to operators and by-standers. Standard fatigue tests are performed for standard specimens; hence, there is an even greater interest in understanding the failures of actual machine components in real working conditions.

In order to test specimens in the laboratory under actual conditions, suitable arrangements, potentially comprising a great number of bodies, would need to be designed. During the operation of equipment, energy is transferred between components via their respective interactions. Some components store energy (stiffness) while others release it (mass/inertia). As a result, in addition to the designed motion, vibrations may occur, applying added load cycles to the system. At specific excitation frequencies, these can lead to anomaly or off-design behaviour and, in the worst case, failure. Although energy dissipation reduces machine efficiency, it also reduces vibration amplitudes depending on the system damping level, which is typically inherently low in torsional systems. Low torsional damping levels lead to high stresses, inducing fatigue equipment failures [9].

As a result of these factors, dynamic interactions play an active role in fatigue crack initiation and growth in real life operation of torsional components. This scenario is often over-looked by

standard fatigue studies, especially under rotating conditions. There is growing interest in the study of fatigue from a dynamic point of view, to better understand the relevant interactions [10, 11, 12] and to reduce experimental investigation time with the use of vibrational excitation [13]. Despite development in this field, studies to date have focused on axial vibration. The need to better understand the impact of torsional vibration on component fatigue life is therefore the motivation of this thesis.

1.2   Previous research on torsional fatigue rigs

Various methods have been proposed for torsional fatigue testing. Examples include hydraulic systems, motor-driven system with external torque control of the input and/or output, ultrasonic torsional methods and complementary back-to-back arrangements. A review of some examples will be presented.

Torsional hydraulic systems represent a traditional solution [14, 15], typically suitable for non-rotating tests (low cycle fatigue), low frequency testing, from 10 Hz to 50 Hz, and high loads (fracture mechanics). This type of arrangement does not replicate real life operation in a representative way. Experimental construction of the S-N diagrams requires long test times, comprising many man-hours, even while working at maximum load frequencies. Hydraulic systems also often require sophisticated control systems. Thus, these systems have their limitations.

Motor-driven systems, which incorporate feedback control on the external motor/brake, although potentially offering greater flexibility in cycle profile design, dissipate, by definition, large amounts of energy during operation. As a result, they also require large amounts of energy to operate. In the absence of sophisticated energy recovery means, such designs thus incur expensive running costs [2] and, with today's heightened awareness of energy wastage, are not be viewed favourably.

Recently, ultrasonic torsional fatigue arrangements have been used to excite specimens with a pure torsional vibration mode at a frequency of 20 kHz [16]. These new mechanical devices can test specimens up to 10^9 - 10^{10} cycles; however, the device transforms an axial mode to an alternating torsional mode. To date, they appear to have been applied to only standard (small) samples and not for actual machine components. Other works relating to very high cycle fatigue (VHCF) using ultrasonic methods are presented by Stanzl-Tshegg *et al.* [17] and Mayer [18, 13].

So as to understand the potential of back-to-back fatigue test rigs, it is appropriate to give a brief review of back-to-back systems in general. Mihailidis and Nerantzis [19] have recently reviewed several mechanical back-to-back devices developed for loading of gears, for which they
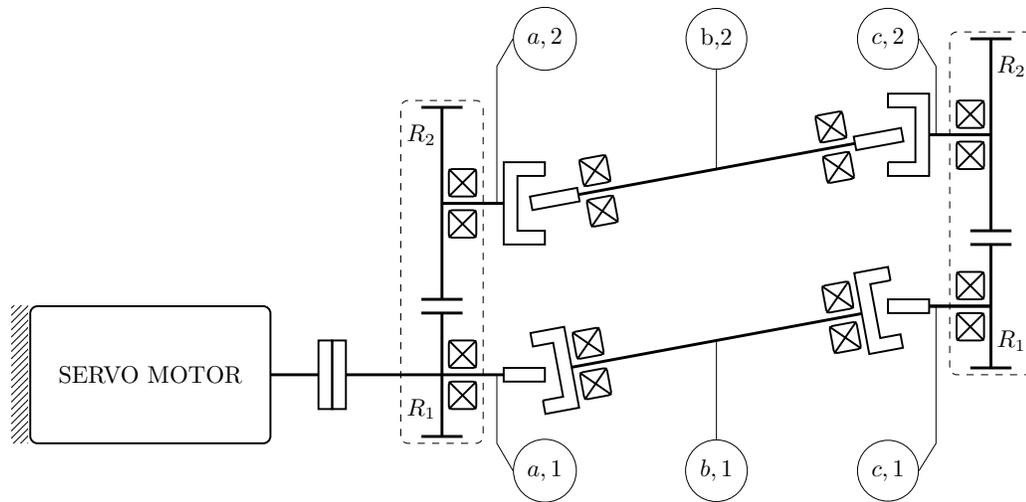
Figure 1.1: Test rig schematic with relevant notation: single back-to-back system [1, 22].

give an adequate explanation. A back-to-back system consists of a gearbox pair connected via a pair of parallel shafts. It is typically used for testing actual gearboxes under load [5, 20, 21]; however, with suitable changes (e.g. the adoption of universal joints) [1], it can also perform torsional fatigue tests on different mechanical components, such as Oldham joints and splined shafts. Such machine components are incorporated in one of the inclined shafts, (b,1) or (b,2), in Fig. 1.1. Previously, a similar scheme was used by Fischer *et al.* [22] to experimentally measure the magnitude of the intermediate-joint axial forces on a driveline transmitting torque at high speed.

As with a parallel back-to-back arrangement, through the power recirculation due to the pre-load, the servomotor needs only to maintain constant angular motion, overcoming the friction torques within the rig. Consequently, small servomotors can be employed to carry out torsional fatigue tests in short times [1]. Furthermore, due to minimal effective resistance torque, the servomotor can be exploited to perform frequency variations of cyclic stress, once again simulating working conditions. The response speed of the multi-body system depends on the dynamic properties of the components.

For a servomotor speed of 3000 rpm, these types of rig allow high loading frequencies of ~100.0 Hz considering that gearbox ratios are set to 1 and the loading cycle is applied twice for each shaft revolution due to the incorporation of the universal joints. Construction simplicity and low manufacturing and operating expense justify the appropriateness of the single back-to-back rig presented by Guzzomi *et al.* [1]. In contrast, the dependency of the amplitude value on the mean value limits experimental construction of S-N curves. Consequently, Guzzomi *et*
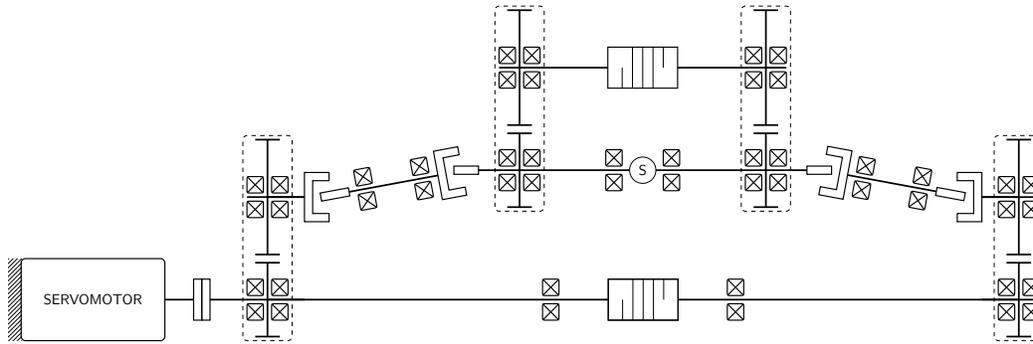
Figure 1.2: Test rig schematic: double back-to-back system [2].

al. developed a double back-to-back arrangement, illustrated in Fig. 1.2 [2]. The torsional stress cycle acting on the specimen Ⓢ results from the combination of two components which can, in theory, be independently set: one the average and the other the oscillating component. Setting the amplitude of the oscillating component is achieved by applying a locked-in moment to the lower back-to-back system via an external pre-load (the first degree-of-freedom (DOF)). The power recycling loop consists of a pair of gearboxes connected by both a multi-component parallel shaft and a pair of non-homokinetic double universal joint drivelines that are attached symmetrically to a second back-to-back system (upper loop). As a result of the non-homokinetic layout, the properties of the non-linear universal joint are emphasised so that a variable torsional stress is passively produced in shaft Ⓢ with each revolution. Including the second back-to-back system in the first closed loop, Guzzomi *et al.* added an extra DOF to the rig through which a shift in the average value of the torsional stress cycle could be achieved via an additional external pre-load (second DOF).

As noted by Guzzomi *et al.*, there appears to be little research in the literature regarding torsional fatigue testing of actual machine components, particularly under rotating conditions [2]. Furthermore, there also appears to be a shift towards implementing advanced control methods for the excitation or reduction of vibration; few studies today focus on passive methods. Of the rig methods proposed in the literature, the back-to-back arrangements presented by Guzzomi *et al.* [2] have the potential to incorporate actual machine test components under rotating conditions. Being passive, they offer potential for energy efficient operation with reasonable test times, while permitting variations in amplitude and mean stress values.

It is clear that, in order to generate the oscillating torsional cycle in both rigs, Guzzomi *et al.* exploited the non-linear velocity transmission characteristic of the universal joint. Extensive works are reported in the literature regarding this joint. A brief review of the salient literature

on the dynamic stability of a rotating shaft and a drive system containing universal joints has been given by Mazzei *et al.* [23]. In this work, the authors focused on the lateral instability of the driven shaft connected by a universal joint to the driving shaft. The investigation of Sheu *et al.* into the effects of joint angles and joint friction in a double universal joint system [24] highlights some interesting phenomena. Using Rayleigh beam theory and including the cross-pin in their dynamic model of a universal joint, they determine the axial torque influence on the critical speed values for the intermediate shaft and, hence, the dependence of axial torque on the viscous friction level and joint angles. Porat [25] developed a general theory for an arbitrary static universal joint modelled as a two-inertia (input-output), masslesss cross-piece with no friction forces in the cylindrical pairs. Yang and Zhishang [4] provided a dynamic dual analysis for a spherical four-link mechanism, which collapsed into a universal joint when given special link proportions and joint constraints. Both transient and steady state cases can be investigated through their formulation, though their model lacks the necessary input and output torques to render it useful as building block element in a dynamic model. Fischer and Freudenstein [26] modelled a universal joint with manufacturing tolerances as a spherical four-link mechanism, developing a dual static analysis. Results permitted the determination of optimum tolerances in the design stage. Using Yang's dual dynamic equation [27], Chen and Freudenstein extended their work to a dual dynamic analysis. Their model could then predict dual bearing forces under high-speed operation. Despite these advances, no mention of contact stiffness or backlash in the kinematic pairs of a universal joint has been found. Thus, it would appears that, although using universal joint non-linear properties to induce a torsional loading cycle on a specimen included in a back-to-back system technically results in an elegant solution, a more in-depth analysis into its non-linear behaviour must be undertaken, both in the frequency and time domains. Such approaches must also focus on developing models of the joint that can be used as building blocks in the multi-body dynamic model of the rigs.

In this context, the present dissertation is concerned with innovative applications of the universal joint as a torsional excitation device and its implementation in the passive back-to-back torsional fatigue test rigs of Guzzomi *et al.* [1, 2]. In particular, the thesis addresses the dynamics of both the joint and the rigs, so as to gain a better understanding of the physical phenomena and to optimise their mechanical design.

1.3   Thesis content

The work presented in this dissertation has been divided into four chapters. A brief description of each is given below:

Chapter 2 presents the receptance theory, a frequency-based sub-structuring technique. Starting from essential concepts, the chapter develops dynamic models of complex systems, consisting of multi-loop components, by combining their sub-system receptances. The described technique also allows displacements at each sub-system coordinate to be predicted so that deflected shapes of the multi-body system at a given frequency can be simulated. Internal forces acting at the extremities of each system component can also be determined.

Chapter 3 presents the torsional receptance models for the components that comprise both passive torsional test rigs under investigation. Due to the central role played by the universal joint in passively producing the torsional stress cycle with each rig, more in-depth analyses are undertaken, both in the frequency and time domains. Two rigid body models for the universal joint are presented: the first comprised of two inertias (input-output) and the second of three inertias (input-floating-output). Equivalent inertias of both models are found to display similar attributes to a second order cosine. For the two-inertia model, the predicted equivalent inertia non-linearity over one revolution is compared with experimental data, resulting in good agreement. Furthermore, due to the similar equivalent inertia variation of both the universal joint and the reciprocating engine in the range $0°$-$360°$, investigations into the dynamic behaviour of a combined system (universal joint $+$ engine) are undertaken and the frequency spectra discussed. Subsequently, the model of Yang and Zhishang for spherical four link mechanisms [4] is adapted by adding a non-zero torque to the dual component at the system output axis, leading to a reformulation of the equations of motion. The advanced three-inertia model for a universal joint is then derived as a specific case and its receptances established. By setting the tensor of inertia for the floating element to zero, the advanced model is reduced to the two-inertia model.

Chapter 4 describes both of the propose passive back-to-back torsional fatigue test rigs (Fig.s 1.1,1.2) in detail. A simple torsional static model of the single-degree-of-freedom (SDOF) back-to-back arrangement is derived and, for different universal joint configurations, the torsional loading cycles are predicted. Subsequently, using the receptance method, a dynamic model of the multi-body apparatus, consisting of lumped-mass and continuous sub-systems, is developed and investigated in the frequency domain. Simulations of Frequency Response Functions (FRFs), both magnitude and phase, and system deflected shapes at damped resonance frequencies are

derived. They are shown to be in good agreement with published results in the literature. The investigation further focuses on the double back-to-back system of which a torsional frequency based model is achieved by adding discrete components and continuous shafts. Receptances over the range 0-1600 Hz have been simulated and twelve resonance frequencies found. Finally, by predicting the angular displacements at coordinates throughout the system at resonant frequencies for suitable boundary conditions, mode shapes have been constructed, disclosing the dynamic response complexity of such a system. Once again, the receptance technique has been proved suitable for the modelling of multi-degree-of-freedom systems.

Chapter 5 summarises the principal findings and suggests new directions for future works.

2 | Receptance Method

'Cognitionis autem duæ sunt species: altera quidem vera & germana, altera verò tenebricosa. Et tenebricosæ quidem sunt hæc omnia, visus, auditus, gustus, tactus. Vera autem & germana est, quæ est ab ea secreta [...][28] '

— Sexti Empirici (c. 160-210 AD)

This chapter recalls some essential concepts of the frequency-based receptance method, integrating relevant computer program scripts developed as part of the present work. In addition, the generalised algorithm for the dynamic modelling of beam-type structures [29] has been comprehensively developed and extended for the torsional rigs investigated in Chapter 4. Computational routines, specifically designed for the connection of mono-dimensional systems with two-coordinates, have been utilised to model architectures that include multi-loop closure schemes, thus extending previous works [5][20].

2.1 Introduction

For dynamic analysis, the receptance technique established by Bishop and Jonhson in 1960 [30] allows steady state vibration characteristics of complex systems to be predicted in the frequency domain. The development of a similar method in the time domain has been proposed by Li *et al.* [31, 32].

The principle steps of the procedure consist of system reduction into smaller components, derivation of the separate solutions for these sub-parts, then coupling of these individual characteristics via suitable conditions. The receptance method is therefore a sub-structuring approach or, more generally, a specific case of *domain decomposition* [33]. To aid with the understanding of the approach, some properties of the technique will be briefly discussed.

The division of the original system into smaller sub-parts inherently simplifies the problem by

defining the component number[1], the required background and the degrees-of-freedom (DOF). Furthermore, it facilitates the building and expansion of shared dynamic model libraries conferring flexibility and speed in the modelling process.

According to specific theories, for example those in Ref.s [34, 35], the sub-structure modelling approach allows insight into the underlying physical phenomena. The formulation of single sub-system models can therefore present great accuracy. Parameters affecting vibrational behaviour, including damping sources[2], can be identified and analysed in order to ascertain their dynamic impact. This leads to a local optimisation of components, which affects the global analysis, reducing computational time.

Experimental sub-system models can be developed and included in the receptance model of a system.

Both local and global checking processes can be performed; the method therefore facilitates detection of problems in component models before their implementation takes place or *in situ* testing of the component is undertaken.

Deflected shapes at any instant are naturally derived, in addition to the mode shapes corresponding to each system natural frequency.

Despite the fact that receptance models of sub-system components can apparently show non-linear behaviour, the procedure requires the use of time-invariant and linear sub-structures with constant parameters throughout the rebuilding stage of the entire system [3].

Construction of a component theoretical model can involve considerable time; however the method permits the easy interchange of sub-system components without complete rederivation of the dynamic equations of the entire system.

2.2 Receptance definition

Consider a linear time-invariant system of which the dynamic steady state behaviour is under investigation in the Fourier domain. The receptance, α_{pq}, can be found by specifying an output response of the system at coordinate p, $u_p e^{j\omega t}$, and measuring or modelling the input excitation required at coordinate q, $F_p e^{j\omega t}$, to produce the output. The receptance is then defined as the ratio of the generalised response to the generalised excitation. Frequency response, compliance and admittance [36] are equivalent terms.

As the linear system assumption enables the use of the superpositioning principle, the equa-

---

[1]project number if one considers a different scale of investigation, e.g. an aircrafts.
[2]viscous/hysteric both concentrated and/or distributed
[3]... and *no internal energy sources* [29]

tions of motion of a n-coordinate system can be expressed, in the subsidiary domain, by:

$$\boldsymbol{u} = |\boldsymbol{\alpha}| \boldsymbol{F} \tag{2.1}$$

Or in matrix form by:

$$
\begin{Bmatrix} u_0 \\ \vdots \\ u_p \\ \vdots \\ u_n \end{Bmatrix}
=
\begin{vmatrix}
\alpha_{00} & \cdots & \alpha_{0p} & \cdots & \alpha_{0n} \\
\vdots & \ddots & \vdots & & \vdots \\
\alpha_{p0} & \cdots & \alpha_{pq} & \cdots & \alpha_{pn} \\
\vdots & & \vdots & \ddots & \vdots \\
\alpha_{n0} & \cdots & \alpha_{nq} & \cdots & \alpha_{nn}
\end{vmatrix}
\begin{Bmatrix} F_0 \\ \vdots \\ F_q \\ \vdots \\ F_n \end{Bmatrix}
\tag{2.2}
$$

In accordance with Maxwell's reciprocal theorem, the receptance matrix $|\boldsymbol{\alpha}|$ results symmetric [30].

2.3 Connection of systems

In the assembly step, the method allows the resultant receptances of the combined system to be derived from the receptance functions of its components. The described process connects mono-dimensional systems characterised by two nodes; however, the procedure can be extended to join three-dimensional bodies of N coordinates. A schematic element representation is shown in Fig. 2.1a. Block A can be joined to a system via coupling point 0, *left port*, and can receive a subsequent component via coupling point 1, *right port*. This refers to a local labelling notation of sub-system coordinates. Fig. 2.1b schematically depicts a complementary singly linked list node
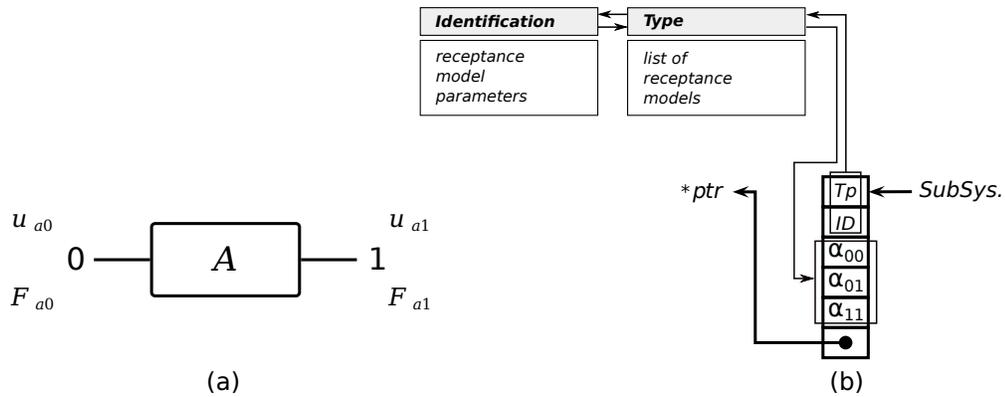
Figure 2.1: Representation of a system: block diagram (a); singly linked list node designed for modelling mono-dimensional components by two coordinates (b).

employed to model a general two-coordinate component in the source code developed in the

present work. The Tp data selects a specific system model type from the list and the ID data then provides the parameters that fully define the transfer functions of the component. Consequently, when the program calls the element, the α_{pq} functions point to the desired receptances. The C structure [37, 38] for a single node is as follows:

```
struct SubSys{
int type;
int ID;
double complex (*a11)(int, double);
double complex (*a12)(int, double);
double complex (*a22)(int, double);
struct SubSys *ptr;
};
```

According to a coordinate numeration of a complete the system that starts from zero, the connecting routine always sets the sub-system ID value to the left coordinate number of sub-system. The element is then identified with respect to its position in the global system. This leads to a second labelling notation, suitable in the addition step, as illustrated in Fig. 2.2a.
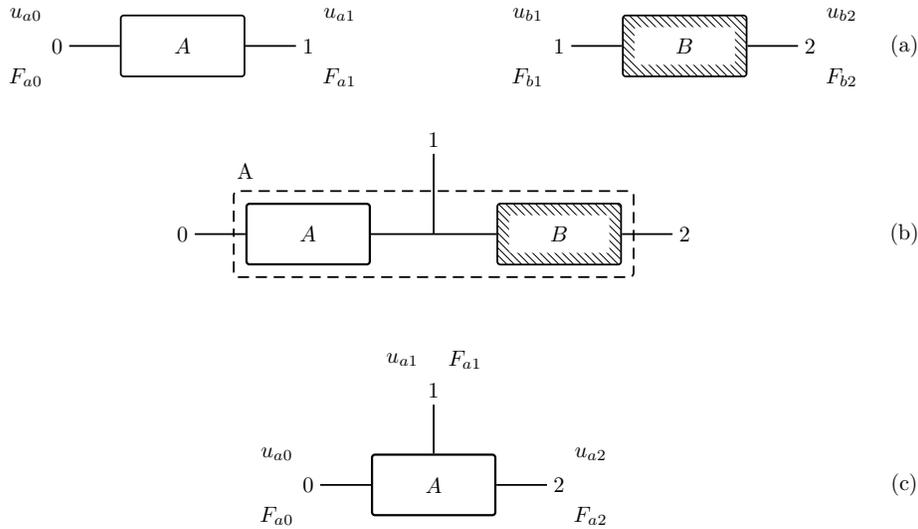
Figure 2.2: General technique for connecting systems.

Applying Equation (2.2) to both systems A and B of Fig. 2.2a yields:

$$\left\{ \begin{array}{c} u_{a0} \\ u_{a1} \end{array} \right\} = \left| \begin{array}{cc} \alpha_{00} & \alpha_{01} \\ \text{SYM} & \alpha_{11} \end{array} \right| \left\{ \begin{array}{c} F_{a0} \\ F_{a1} \end{array} \right\} \qquad\qquad \left\{ \begin{array}{c} u_{b1} \\ u_{b2} \end{array} \right\} = \left| \begin{array}{cc} \beta_{11} & \beta_{12} \\ \text{SYM} & \beta_{22} \end{array} \right| \left\{ \begin{array}{c} F_{b1} \\ F_{b2} \end{array} \right\} \qquad (2.3)$$

Where the α_{pq} and β_{pq} functions are known.

Table 2.1: Receptances of a combined system. $\Delta = 0$ is the frequency function [30].

SYSTEM	RECEPTANCE		
	α_{00}	α_{11}	α_{22}
	$\alpha_{00} - \dfrac{\alpha_{01} \cdot \alpha_{10}}{\Delta}$	$\dfrac{\alpha_{11} \cdot \beta_{11}}{\Delta}$	$\beta_{22} - \dfrac{\beta_{21} \cdot \beta_{12}}{\Delta}$
	$\alpha_{01} = \alpha_{10}$	$\alpha_{12} = \alpha_{21}$	$\alpha_{02} = \alpha_{20}$
$\Delta = \alpha_{11} + \beta_{11}$	$\alpha_{01} - \dfrac{\alpha_{01} \cdot \alpha_{11}}{\Delta}$	$\beta_{21} - \dfrac{\beta_{21} \cdot \beta_{11}}{\Delta}$	$\dfrac{\alpha_{01} \cdot \beta_{12}}{\Delta}$

Consider the balance and compatibility equations of coupling point 1, Fig. 2.2b:

$$F_{a1} + F_{b1} = F_1 \tag{2.4}$$

$$u_{a1} = u_{b1} = u_1 \tag{2.5}$$

Rearranging Equations (2.3) to (2.5) gives the receptances of the built-up system, Fig. 2.2c, as a function of its components, α_{pq} and β_{pq}. Table 2.1 summarises the results [30]. Each receptance is consequently characterised by the same denominator, Δ, which is completely defined by the direct receptances of the connected systems at the coupling point:

$$\Delta = \alpha_{11} + \beta_{11} \tag{2.6}$$

When set to zero, Δ is known as the *frequency equation*, the solutions of which yield the resonant natural angular frequencies, ω_n, of the entire system.

A further analysis of the results in Table 2.1 enables a rapid receptance derivation approach. In fact, the α_{pq} formulation depends on the relative positions of nodes p and q with respect to the system coupling point 1. Therefore, when p and q are on the opposite sides of 1 , i.e. $[p,1,q]$ ($[q,1,p]$):

$$\alpha_{pq} = \frac{\alpha_{p1} \cdot \beta_{1q}}{\Delta} \tag{2.7}$$

And in other cases:

$$\alpha_{pq} = \chi_{pq} - \frac{\chi_{p1} \cdot \chi_{1q}}{\Delta} \qquad \text{where } \chi = \begin{cases} \alpha & \text{for} \quad [1,p,q] \text{ or } [1,q,p] \\ \beta & \text{for} \quad [p,q,1] \text{ or } [q,p,1] \end{cases} \tag{2.8}$$

The rational term includes the dynamic information of the added element via coupling point 1. Using Equation (2.6) and noting the sequence of numerator subscripts: response, coupling,
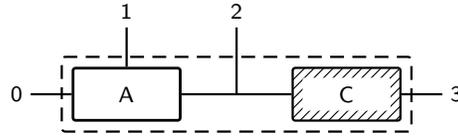
Figure 2.3: Effect of increasing coupling point due to the connection.

excitation; its derivation becomes straightforward. Some α_{pq} functions belong to both groups, such as α_{01}, α_{11} and α_{12}, which is a consequence of Equation (2.5).

The connection produces an increase in the number of *right port* coordinates; however, only one is used for adding the next element, as shown in the example presented in Fig. 2.3. The unattached coupling points must be carried forward if the modelling is to allow them to engage a new component at later stages. This involves an increase in the size of the receptance matrix of the under-construction system; however, this aspect has not been developed in the current thesis. When the coordinate number of the transforming system is held steady, the routine handling the sub-system connections becomes trivial; the added elements are ordered in sequence. Fig. 2.4 shows the data structure schema.
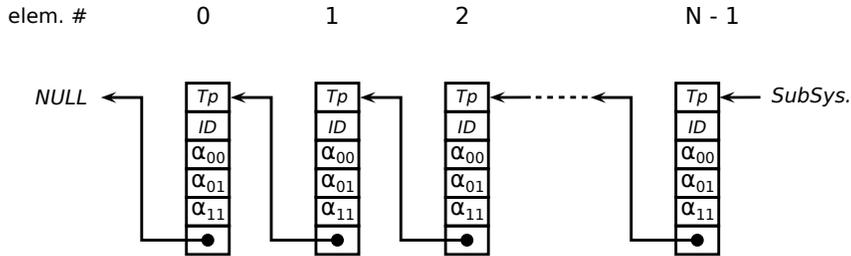
Figure 2.4: A singly linked list schema: N components.

C routines developed to compute the direct receptance of the maximum coordinate N, α_{NN}, and each cross receptance, α_{pq}, are as follow:

```
/* DIRECT-RECEPTANCES */
double complex Direct(struct SubSys *s, double w)
{struct SubSys *s0 = s->ptr;
 double complex a00, a01, a11;
 double complex DRC;

 a00=s->a00(s->ID, w);
 a01=s->a01(s->ID, w);
 a11=s->a11(s->ID, w);

 if(s0 != NULL)
  DRC = a11 - cpow(a01,2)/(a00 + Direct(s0, w));
 else
  DRC = a11;
 return DRC;
```

```
}
/* CROSS-RECEPTANCES */
double complex CROSS(struct SubSys *p,double w,int theta,int Torque)
{struct SubSys *p0=p->ptr;
 int _MAX_Node;
 int th,T;
 double complex rtn;

 _MAX_Node = MAX_COORD(p);
 th = max(theta,Torque);
 T = min(theta,Torque);

 if(!p0){
   if(T == p->ID){
     if(th  == T) rtn = p->a11(p->ID,w);
     else if (th == _MAX_Node) rtn = p->a12(p->ID,w);
   }
   else if (T == _MAX_Node && th == T ) rtn = p->a22(p->ID,w);
 }
 else{
   if(th == _MAX_Node){
     if(T == th) rtn = Direct(p,w);
     else  rtn = p->a12(p->ID,w)*CROSS(p0,w,p->ID,T)
             /(p->a11(p->ID,w)+CROSS(p0,w,p->ID,p->ID));
   }
   else if(th == p->ID)
     rtn = p->a11(p->ID,w)*CROSS(p0,w,p->ID,T)
         /(p->a11(p->ID,w)+CROSS(p0,w,p->ID,p->ID));
   else  rtn = CROSS(p0,w,th,T)
           - CROSS(p0,w,th,p->ID)*CROSS(p0,w,p->ID,T)
             /(p->a11(p->ID,w) + CROSS(p0,w,p->ID,p->ID));
 }
 return rtn;
}
```

2.4   Loop closure and multi-loop schema

The standard construction process of an open gearbox loop by contiguous sub-system addition has been well described by Leishman *et al.* [5], Fig. 2.5a-b. The transformation from a three-coordinate open system, B, into a two-coordinate closed system, A, has also been formulated by the same authors using a classical approach [5]. It results in the correspondence of the new *left port*, i, with the connected nodes, $2 \cup 3$, Fig. 2.5b-d. The unknown receptances, α, of the new system are consequently expressed in terms of β. Leishman *et al.* applied the procedure to dynamic modelling of a back-to-back system commonly employed for the investigation of gearbox dynamics. This back-to-back system consists of a pair of gearboxes, $GB_{L/R}$, connected with two parallel shafts, ① and ②, forming a closed loop, Fig. 2.6.

In the construction of a back-to-back component, however, the i-node can be physically considered equivalent to the *right port*, 1, because of the element addition sequence starting with a gearbox sub-system, Fig. 2.5e. The choice of a gearbox sub-system as the first element has some advantages. In fact, a gearbox should be correctly modelled by a three-coordinate system, the dynamics of which is completely defined by α_{11}, α_{22}, α_{33}, α_{12}, α_{13} and α_{23}; however, a
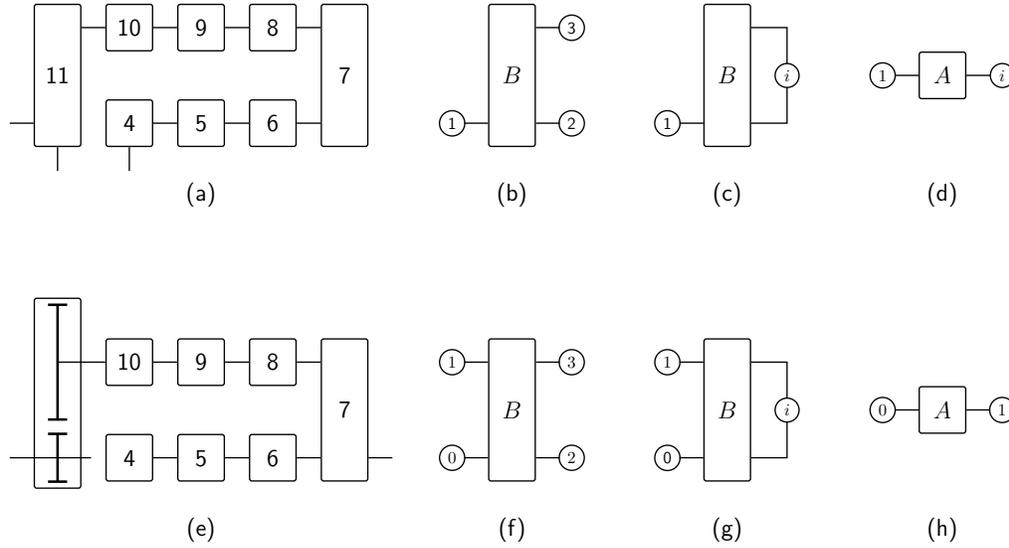
Figure 2.5: Addition sequence of elements starting with a general three-coordinate system, 11, (a); transformation of a three-coordinate open system, B, into a two-coordinate closed system, A (b) to (d) [5]; an alternative model of a closed-loop component, well suited to multi-loop schema architectures (e) to (h).

two-coordinate system is sufficient. Referring to Fig.s 2.5b-e, the kinematics of coordinates 1 and 2 are equivalent, because both refer to the same inertia. Therefore, this consideration gives $\alpha_{11} = \alpha_{12}$, $\alpha_{21} = \alpha_{22}$ and $\alpha_{13} = \alpha_{23}$.

As a result, the coupling of a closed-loop system, reduced into a two coordinate system, becomes a composite structure that is a *peninsular* in form; that is, no further elements can be attached. A priori, such a component makes it impossible for the model to receive extra components on this side. In applications for which closed sub-systems with distinct left-right ports must be modelled, it is therefore necessary that the number of coordinates in the system must be reduce by at least two, Fig. 2.5f-h.

The receptances of the mono-dimensional transformed system are:

$$\alpha_{pq} = \beta_{qs} + \frac{(\beta_{p2} - \beta_{p3})(\beta_{3q} - \beta_{2q})}{\Delta} \tag{2.9}$$

$$\alpha_{ii} = \beta_{23} + \frac{(\beta_{22} - \beta_{23})(\beta_{33} - \beta_{32})}{\Delta} \tag{2.10}$$

Where:

$$\Delta = (\beta_{22} - \beta_{23}) + (\beta_{33} - \beta_{32}) \tag{2.11}$$

Equation (2.10) has been included for completeness due to the choice of first element.
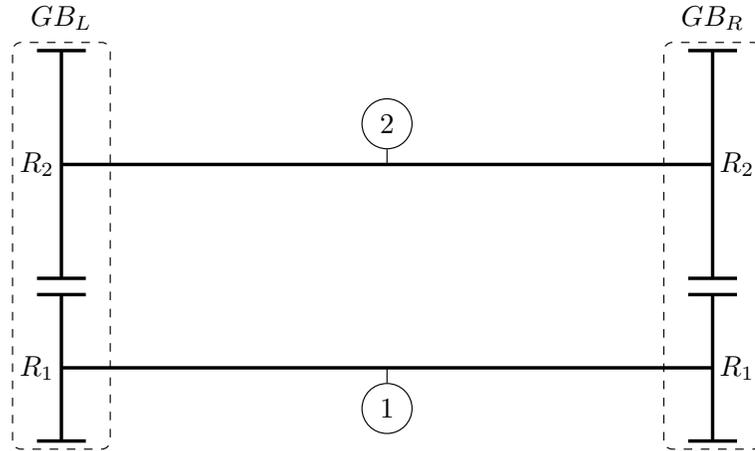
Figure 2.6: Back-to-back simple scheme.

A rapid α_{pq} derivation procedure may even be performed in this case by extending Equation (2.8) to equivalent terms. Referring to Equation (2.9), the rational component adds the dynamics of the transformed system to the previous contribution β_{qs}. Implementing a closed loop involves connecting two points of the same structure (i.e. 2 and 3). The added information therefore integrates the dual constraint with the differences in receptances, $\beta_{s2} - \beta_{s3}$. Attention should be paid to the sequence of numerator subscripts in Equations (2.9) and 2.10 in correspondence with the components in Fig. 2.5. Finally, Δ, expressed by Equation (2.11), can be seen as the sum of equivalent direct receptances of the engaging points.

Because of the multi-body nature of the element, the *left* and *right ports* are not unique and must be provided as inputs to the program; however, the closing points, (e.g. 2 and 3), are explicitly defined by the loop construction procedure.

The receptances of a mono-dimensional, two-coordinate closed loop system are stored in a singly linked list node, Fig. 2.1b, and added to an appropriate list, called *primary*-list, according to Subsection 2.3. Consequently, at such a level, the node call implies that the three α_{pq} functions point to Equation (2.9). A type-number, identifying the element inside the developed program, is assigned to each component. Type-67 corresponds to a closed-loop component.

The components in an open loop system are managed as described in Subsection 2.3 and then allocated to a new list, as in Fig. 2.4, called the *secondary*-list.

In systems that include this type of element, α_{00}, α_{01} and α_{11} may not be enough to completely define the dynamic behaviour. Selecting its internal nodes for dynamic investigation, as well as for excitation of this structure, requires more receptances of the closed loop to be calculated. Following the rules in Table 2.1 implies that, for a single node, s, two frequency

responses relating s to the external ports of the element, 0 and 1, are required: α_{0s} and α_{s1}. For the case of two internal nodes, p and q, five such responses are instead required: α_{0p}, α_{p1}, α_{0q}, α_{q1} and α_{pq}.

The aforementioned approach for closed-loop components is well suited to the modelling of architectures consisting of multi-loop schema. As the element is basically a two-coordinate sub-system, adding a new type-67 to the same *primary*-list of components results in a linear layout, while adding a new type-67 to a *secondary*-list results in a nested layout (a loop inside a loop). Therefore, with L number of closed-loop components, the routine must manage at least one *primary*-list and L *secondary*-lists.

2.5   Deflected shapes and internal strains

In order to investigate the deflected shapes of a multi-body system for a given frequency, $\overline{f}$, two approaches are presented. Both methods use a coordinate displacement, u_p, set to 1.0, and find the excitation, F, acting at coordinate q, required to produce the imposed displacement, calculated via the known receptance, α_{pq}, Equation (2.12):

$$u_p = 1.0 \qquad\qquad \Rightarrow \qquad\qquad F_q = \frac{u_p}{\alpha_{pq}} = \frac{1.0}{\alpha_{pq}} \qquad\qquad (2.12)$$

The procedure imposes unity displacement at the end-coordinate of the main sub-system list; however, alternative choices can be made depending on the particular investigated multi-body system[4].

As every system receptance, α_{kq}, can be now estimated using Equation (2.12), the first method permits prediction of each displacement of the system, u_k, as follows:

$$u_k = \alpha_{kq} \cdot F_q = \alpha_{kq} \cdot \frac{1.0}{\alpha_{pq}} \qquad\qquad (2.13)$$

As u_k depends on α_{kq} of the complex system, the first technique, formulated by Equation (2.13), can be used to check the reliability of the source code. Specifically, the results are compared to the displacements arising from a second approach, which is based on a different method.

In this second method, again using Equation (2.12), the operational conditions are defined, in particular the external force vector. The complex system is then broken-down, one component at a time, starting from one extreme of the main list. Through the detached sub-system receptances

---

[4]For example, with regard to the modelled passive torsional rig, the topic of this thesis dissertation, an angular displacement set to 1.0 corresponds to the output servo-motor inertia coordinate, θ_1, which is an internal coordinate. The torque excitation is also applied at the same node.

and the balance equation of the coupling co-ordinate, excitations acting at the sub-system *left* and *right ports* can be calculated. Using Equation (2.1) the unknown displacement is found. As a result, a recursive process can be employed. A basic example is depicted in Fig. 2.7a. Let the displacement of coordinate 0 be constrained such that $u_0 = u_{b0} = 1.0$ and let coordinate 2 be selected as the exciting node for the system in order to produce a fixed u_0. Therefore:

$$F_2 = \frac{1.0}{\alpha_{02}} \tag{2.14}$$

There are no other forces acting on the system, so $F_0 = F_{b0} = 0.0$. The breaking-up phase of

Figure 2.7: Derivation of displacement and internal forces via detachment process.

A sees sub-system B detached from coordinate 1. The dynamic characteristics of B, β_{00}, β_{01} and β_{11} are well established. Consequently, F_{b1} can be determined:

$$F_{b1} = \frac{u_0 - \beta_{00} F_{b0}}{\beta_{01}} \tag{2.15}$$

Applying Equation (2.1) to sub-system B gives:

$$u_{b1} = \beta_{10} F_{b0} + \beta_{11} F_{b1} \tag{2.16}$$

according to the compatibility equation, $u_{b1} = u_{c1} = u_1$.

Finally the balance equation for coordinate 1 yields

$$F_{c1} = F_1 - F_{b1} \tag{2.17}$$

The aforementioned steps can be performed recursively for every subsequent component, so as to attain the deflected shape and the internal strains of the system for a specific frequency.

The presence of a closed loop sub-system requires a further in-depth analysis. Firstly, treating it as a two-coordinate element, the displacements and the strains of the external connections (i.e. u_{b0}, F_{b0} and u_{b1}, F_{b1} relative to Fig. 2.5h) can be easily achieved. Secondly, by dividing point i into 2 and 3, Fig. 2.5g - f, F_{b2} can be derived. In fact, considering:

$$u_0 = \beta_{00} F_{b0} + \beta_{01} F_{b1} + \beta_{02} F_{b2} + \beta_{03} F_{b3} \tag{2.18}$$

And:

$$F_i = F_{b2} + F_{b3} \tag{2.19}$$

Rearranging gives:

$$F_{b2} = \frac{u_0 - \beta_{00} F_{b0} - \beta_{01} F_{b1} - \beta_{03} F_i}{\beta_{02} - \beta_{03}} \tag{2.20}$$

Every term on the right hand side of Equation (2.20) is known, including, F_i, as a result of Equation (2.12), which defines the external force vector acting on the system. In order to re-apply the described routine to the closed-loop component list, Equations (2.15) to (2.17), T_{b3} and u_2 must be made explicit. As Equation (2.19) supplies F_{b3}, it follows that:

$$u_2 = \beta_{20} F_{b0} + \beta_{21} F_{b1} + \beta_{22} F_{b2} + \beta_{23} F_{b3} \tag{2.21}$$

Referring to Fig. 2.7e, there is another element of the open-loop system that needs a separate treatment: sub-system 7, including *left-port* 1 of the closed-loop component. The standard routine, Equations (2.15) to (2.17), cannot merely process this element due to its characteristic. It is a three-coordinate sub-system; however, applying the classical approach to 7 leads to no particular difficulties. Having already detached components 4 to 6 from the open system, the displacement and internal forces of the node coupling 7 and 8 are unknown. Using the local labelling notation, the investigated node refers to 0. Displacements and internal forces at nodes

1 and 3 have been derived in previous steps. Expressing u_{a1} as a function of the forces acting at the nodes of subsystem 7 via its receptances:

$$F_{b0} = \frac{u_{a1} - \beta_{11}F_{a1} - \beta_{13}F_{b3}}{\beta_{10}} \tag{2.22}$$

Then:

$$u_{b0} = \beta_{00}F_{b0} + \beta_{01}F_{a1} + \beta_{03}F_{b3} \tag{2.23}$$

The described approach is mainly based on the sub-system component receptances and not on the receptances of the complex system. Using both techniques, the program yields the same results. Appendix C.1 shows the source code developed to calculate the receptances of multi-body systems consisting of mono-dimensional two-coordinate components, the deflected shapes and the internal forces acting on the sub-system elements.

3 | Component Receptances

'The best known [...] is the Universal Joint. [...] It consists broadly of two end pieces, and a middle piece, the latter containing two pairs of journals placed at right angles with each other in the form of cross, each pair fitting into journals on one and the other of the end pieces respectively [39]. '

– Reuleaux (Set.30, 1829 - Aug.20, 1905)

In this chapter, an investigation into the components required to model passive torsional fatigue test rigs has been undertaken. Having broken up the complicated multi-degree-of-freedom systems into smaller sub-systems, the principal aim of the current chapter is to derive a torsional receptance model for each element. For some of the components, such as the servomotor, the flexible coupling and the gearbox, published models available in the literature have been used and are listed here within for readers' convenience. A particular case concerns the continuous bar with distributed damping. Its torsional receptances (analogous to the axial formulation) have been presented by Derry and Stone [40]; however, to date no derivation of these equations has been published. Consequently, in order to better understand and correctly use these equations, their derivation from the equilibrium of an infinitesimal bar element has been undertaken independently here and resubmitted.

Two novel frequency-based receptance formulations for a universal joint in closed form have further been derived: Section 3.3.1 reports the study regarding the simplest two-inertia model [41]; Section 3.3.2 reports a more advanced three-inertia model. Because of the role of the universal joint in rotating machinery, an in-depth investigation into its equivalent inertia in the frequency and time domains has been carried out in Section 3.3.1. Doing so has suggested a novel application of the component as a passive device for the balance of inertia variation in slider-crank mechanisms and hence application to the reciprocating internal combustion engine.

This chapter also includes an experimental validation of the predicted torsional anti-resonance frequency over one revolution and the subsequent confirmation of its inertia function.

Section 3.3.2 presents an advanced model of a universal joint as a special case of a spherical four-link mechanism. Such a model permits the inclusion of mass/inertia effects. As a result, vibrational sources due to the out-of-balance forces arising from the floating element mass and inertia can be estimated, including the relative gyroscopic effects. Although the investigation has confirmed small variations between the results of the simple two-inertia model and those of the more advanced model, contact effects such as friction, stiffness and play could potentially be introduced in the latter.

3.1 Receptances of standard sub-system components

Table 3.1, below, reports the standard receptances of components that are present in typical back-to-back rigs [5]. The receptances of a servomotor model, characterised by abutment excitation, have also been added to the list.

3.2 Continuous bar with distributed hysteretic damping

In this section, the axial/torsional receptances of a continuous bar with distributed hysteretic damping have been derived. In future works, the receptances of a continuous bar with distributed viscous damping will be undertaken and the results compare to those of Derry and Stone [42].

It is a common practice to ignore the damping characteristic of bars included in mechanical systems. This is because of their small influence when compared to the damping level of the entire system; however, in some cases, such as in torsional vibrations of rotating machines or in the case of non-metallic materials, the dynamic behaviour of system may be significantly affected by damping sources.

The aim of this analysis is to better understand the phenomena, deriving receptance equations that have not been published. The investigation concerns the axial vibrations of a free/free bar excited by an oscillating force acting at the end $x = L$, Fig. 3.1 [30]; however, the final equations can be easily extended to the case of torsional vibration.

Newton's second law, applied to the infinitesimal element in Fig. 3.1, becomes:

$$\rho(x)A(x)dx\frac{\partial^2 u}{\partial t^2} = \frac{\partial N}{\partial x}dx + \frac{\partial H}{\partial x}dx \tag{3.1}$$

Table 3.1: Some subsystem dynamic models from [5]

RECEPTANCES

(a) AC SERVOMOTOR
1) Exciting torque
$\alpha_{00} = \alpha_{01} = \alpha_{10} = 0$
$\alpha_{11} = \dfrac{\theta_1}{\mathsf{T}_1} = \dfrac{1}{\mathsf{k} + \mathsf{j}\omega\mathsf{c} - \omega^2\mathsf{I}_1}$
(2)Exciting abutment
$\alpha_{00} = 1$
$\alpha_{01} = \alpha_{10} = \dfrac{\mathsf{k} + \mathsf{j}\omega\mathsf{c}}{\mathsf{k} + \mathsf{j}\omega\mathsf{c} - \omega^2\mathsf{I}_1}$
$\alpha_{11} = \dfrac{\theta_1}{\mathsf{T}_1} = \dfrac{1}{\mathsf{k} + \mathsf{j}\omega\mathsf{c} - \omega^2\mathsf{I}_1}$

(b) COUPLING
$\alpha_{00} = \dfrac{\theta_0}{\mathsf{T}_0} = \dfrac{(\mathsf{k} + \mathsf{j}\omega\mathsf{c}) - \omega^2\mathsf{I}_1}{\omega^4\mathsf{I}_0\mathsf{I}_1 - \omega^2(\mathsf{k} + \mathsf{j}\omega\mathsf{c})(\mathsf{I}_0 + \mathsf{I}_1)}$
$\alpha_{01} = \alpha_{10} = \dfrac{\theta_1}{\mathsf{T}_0} = \dfrac{(\mathsf{k} + \mathsf{j}\omega\mathsf{c})}{\omega^4\mathsf{I}_0\mathsf{I}_1 - \omega^2(\mathsf{k} + \mathsf{j}\omega\mathsf{c})(\mathsf{I}_0 + \mathsf{I}_1)}$
$\alpha_{11} = \dfrac{\theta_1}{\mathsf{T}_1} = \dfrac{(\mathsf{k} + \mathsf{j}\omega\mathsf{c}) - \omega^2\mathsf{I}_0}{\omega^4\mathsf{I}_0\mathsf{I}_1 - \omega^2(\mathsf{k} + \mathsf{j}\omega\mathsf{c})(\mathsf{I}_0 + \mathsf{I}_1)}$

(c) SPUR GEAR PAIR
$\alpha_{00} = \dfrac{\theta_0}{\mathsf{T}_0} = \dfrac{(\mathsf{k} + \mathsf{j}\omega\mathsf{c})\mathsf{R}_1^2 - \omega^2\mathsf{I}_1}{\omega^4\mathsf{I}_0\mathsf{I}_1 - \omega^2(\mathsf{k} + \mathsf{j}\omega\mathsf{c})(\mathsf{I}_0\mathsf{R}_1^2 + \mathsf{I}_1\mathsf{R}_0^2)}$
$\alpha_{01} = \alpha_{10} = \dfrac{\theta_1}{\mathsf{T}_0} = \dfrac{(\mathsf{k} + \mathsf{j}\omega\mathsf{c})\mathsf{R}_0\mathsf{R}_1}{\omega^4\mathsf{I}_0\mathsf{I}_1 - \omega^2(\mathsf{k} + \mathsf{j}\omega\mathsf{c})(\mathsf{I}_0\mathsf{R}_1^2 + \mathsf{I}_1\mathsf{R}_0^2)}$
$\alpha_{11} = \dfrac{\theta_1}{\mathsf{T}_1} = \dfrac{(\mathsf{k} + \mathsf{j}\omega\mathsf{c})\mathsf{R}_0^2 - \omega^2\mathsf{I}_0}{\omega^4\mathsf{I}_0\mathsf{I}_1 - \omega^2(\mathsf{k} + \mathsf{j}\omega\mathsf{c})(\mathsf{I}_0\mathsf{R}_1^2 + \mathsf{I}_1\mathsf{R}_0^2)}$

Deformation and restoring force are here reported for completeness:

$$\varepsilon = \frac{\Delta l}{l} = \frac{\partial u}{\partial x} \tag{3.2}$$

$$N(x) = E(x)A(x)\varepsilon \tag{3.3}$$

H in Equation (3.1) refers to the damping force. The hysteretic damping for a discrete, lumped-mass system can be expressed as $H = k\eta\backslash\omega \cdot \dot{s}$. Considering:

$$k(x) = \frac{E(x)A(x)}{dx} \tag{3.4}$$

$$\dot{s} = \frac{\partial}{\partial t}(s) = \frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}dx\right) \tag{3.5}$$

The result for a continuous bar is:

$$H(x) = \frac{E(x)A(x)}{dx} \cdot \eta(x) \cdot \frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}dx\right) \tag{3.6}$$
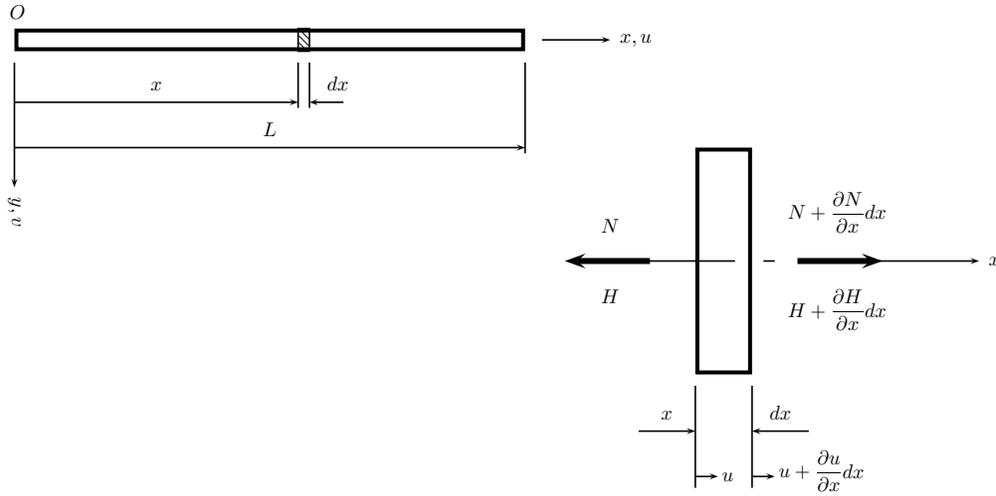
Figure 3.1: Bar of length L and an infinitesimal element thereof.

"The inclusion of $EA\backslash dx$ in the formula (3.6) is justified by the fact that, in this case, [...] the damping is a small additional part of the total reaction of the structure under distortion, of which [...] the structural (hysteretic) damping differs significantly from the viscous one which is quite unrelated to the restoring force" [43]

As a result of this observation, if the viscous damping were considered in place of the hysteretic damping, Equation (3.5) would be $\partial u/\partial t$. Therefore, it would be of interest to develop the receptances of a bar for distributed viscous damping and compare the result to Derry and Stone's model. They state that to do so, one must substitute the non-dimensional η coefficient with $\omega\xi$, where ξ is the damping ratio.

Substituting Equations (3.3) and (3.6) into Equation (3.1), and considering A, E and η as constant values yields:

$$\rho(x)\frac{\partial^2 u}{\partial t^2} = E\left[\frac{\partial^2 u}{\partial x^2} + \frac{\eta}{\omega}\frac{\partial^2}{\partial x^2}\left(\frac{\partial u}{\partial t}\right)\right] \tag{3.7}$$

Assuming that $u(x,t) = U(x)e^{j\omega t}$ represents the steady state motion, this leads to:

$$\frac{\partial u}{\partial t} = j\omega U(x)e^{j\omega t} \qquad\qquad \frac{\partial^2 u}{\partial t^2} = -\omega^2 U(x)e^{j\omega t} \tag{3.8}$$

$$\frac{\partial u}{\partial x} = \dot{U}(x)e^{j\omega t} \qquad\qquad \frac{\partial^2 u}{\partial x^2} = \ddot{U}(x)e^{j\omega t} \tag{3.9}$$

Using Equations (3.8) and (3.9) in Equation (3.7) and rearranging gives:

Axial vibration

$$\ddot{U}(x) + \frac{\omega^2 \rho(x)}{E(1+j\eta)} U(x) = 0 \qquad \left(\begin{array}{l} \text{Torsional vibration} \\[2mm] \ddot{\Theta}(x) + \frac{\omega^2 \rho(x)}{G(1+j\mu)} \Theta(x) = 0 \end{array} \right) \qquad (3.10)$$

where parameter $\eta = h \backslash E$ ($\mu = h \backslash G$) is dimensionless.

The solution of Equation (3.10) for constant ρ is $U(x) = Be^{\phi_1 x} + Ce^{\phi_2 x}$ [44], where:

$$\phi^2 = j^2 \frac{\lambda^2 (\gamma + j\delta)^2}{1+\eta^2} \qquad (3.11)$$

Where:

$$\lambda^2 = \frac{\rho\omega^2}{E} \qquad \gamma = \frac{[(1+\eta^2)^{\frac{1}{2}} + 1]^{\frac{1}{2}}}{\sqrt{2}} \qquad \delta = -\frac{[(1+\eta^2)^{\frac{1}{2}} - 1]^{\frac{1}{2}}}{\sqrt{2}}$$

Using results presented fully in Appendix B.1, ϕ_1 and ϕ_2 can be expressed as follows:

$$\phi_1 = \pm j\lambda \frac{(\gamma + j\delta)}{(1+\eta^2)^{\frac{1}{2}}} \qquad\qquad \phi_2 = \pm j\lambda (R + jS) \qquad (3.12)$$

$$R = \frac{[(1+\eta^2)^{\frac{1}{2}} + 1]^{\frac{1}{2}}}{\sqrt{2}(1+\eta^2)^{\frac{1}{2}}} \qquad\qquad S = -\frac{[(1+\eta^2)^{\frac{1}{2}} - 1]^{\frac{1}{2}}}{\sqrt{2}(1+\eta^2)^{\frac{1}{2}}}$$

Therefore:

$$U(x) = Be^{j\lambda(R+jS)x} + Ce^{-j\lambda(R+jS)x} \qquad (3.13)$$

The constants B and C depend on the end conditions of the bar. If the bar is free/free and excited at the end $x = L$ by a force $F(t) = Fe^{j\omega t}$, the boundary conditions are:

$$\begin{cases} F = 0 & \text{at } x = 0 \\[2mm] F \neq 0 & \text{at } x = L \end{cases} \qquad (3.14)$$

Figure 3.2 shows the balance of internal and external forces at the excited extreme of the bar ($x = L$). It also depicts the equilibrium condition at the same section for discrete lumped-mass
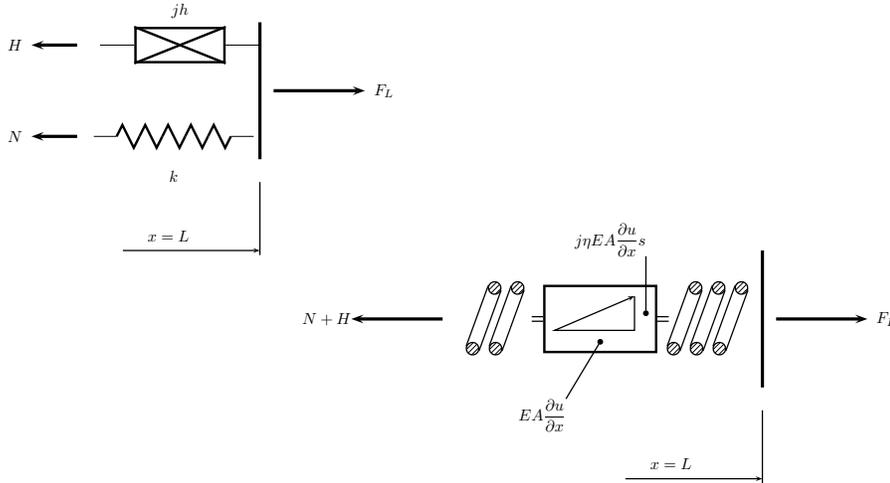
Figure 3.2: Internal and external forces applied to the section at $x = L$ for a discrete, lumped-mass system (top-left) and for a continuous system (bottom-right)

and continuous systems. Substituting Equations (3.3) and (3.6) into Equation (3.14) gives:

$$\begin{cases} EA(1+j\eta)\dfrac{\partial \mathsf{U}}{\partial x} = 0 & \text{at } x = 0 \\[2mm] EA(1+j\eta)\dfrac{\partial \mathsf{U}}{\partial x}e^{j\omega t} = F_L & \text{at } x = L \end{cases} \tag{3.15}$$

The equation for $x = L$ can be re-organising as follows:

$$F_L = \left[EA(1+j\eta)\dfrac{\partial \mathsf{U}}{\partial x}\right]e^{j\omega t} = \mathsf{F_L}e^{j\omega t} \tag{3.16}$$

Its vibrational nature results from the assumptions made about $u(x,t)$. For the current study, it is on interest to find the response term as a function of x. Therefore:

$$\begin{cases} \dfrac{\partial \mathsf{U}}{\partial x} = 0 & \text{at } x = 0 \\[2mm] \dfrac{\partial \mathsf{U}}{\partial x} = \dfrac{\mathsf{F_L}}{EA} \cdot \dfrac{1}{(1+j\eta)} & \text{at } x = L \end{cases} \tag{3.17}$$

Differentiating Equation (3.13) with respect to x results

$$\dfrac{\partial \mathsf{U}(x)}{\partial x} = j\lambda(R+jS)\left[Be^{j\lambda(R+jS)x} - Ce^{-j\lambda(R+jS)x}\right] \tag{3.18}$$

Combining Equations (3.17) in (3.18):

$$\begin{cases} 0 & = j\lambda(R+jS)(B-C) \\[2mm] \dfrac{F_L}{EA} & = j\lambda(R+jS)(1+j\eta)\big[Be^{j\lambda(R+jS)L} - Ce^{-j\lambda(R+jS)L}\big] \end{cases} \tag{3.19}$$

The values of B and C can be derived:

$$\begin{cases} C & = B \\[2mm] \dfrac{F_L}{EA} & = j\lambda(1+j\eta)(R+jS)B\big[e^{j\lambda(R+jS)L} - e^{-j\lambda(R+jS)L}\big] \end{cases} \tag{3.20}$$

Using the first result of Equation (3.20), $B = C$, in Equation (3.13) and rearranging gives:

$$B = \frac{U(x)}{\big[2\cos(\lambda Rx)\cosh(\lambda Sx) - 2j\sin(\lambda Rx)\sinh(\lambda Sx)\big]} \tag{3.21}$$

From the second result of Equation (3.20), B can be expressed explicitly:

$$B = \frac{F_L}{EA} \cdot \cfrac{1}{\begin{Bmatrix} j\lambda(1+j\eta)(R+jS) \\ \cdot\big[-2\cos(\lambda RL)\sinh(\lambda SL) \\ +2j\sin(\lambda RL)\cosh(\lambda SL)\big] \end{Bmatrix}} \tag{3.22}$$

Substituting B of Equation (3.22) into Equation (3.21) and rearranging yields:

$$U(x) = \frac{F_L}{2\lambda EA} \cdot \cfrac{\begin{bmatrix} 2\cos(\lambda Rx)\cosh(\lambda Sx) \\ -2j\sin(\lambda Rx)\sinh(\lambda Sx) \end{bmatrix}}{\begin{Bmatrix} -(1+j\eta)(R+jS) \\ \cdot\big[\sin(\lambda RL)\cosh(\lambda SL) \\ +j\cos(\lambda RL)\sinh(\lambda SL)\big] \end{Bmatrix}} \tag{3.23}$$

Expressing the numerator of Equation (3.23) in its real, a, and imaginary, b, parts:

$$a = 2\lambda EA[(S+\eta R)\cos(\lambda RL)\sinh(\lambda SL) \tag{3.24}$$

$$+ (\eta S - R)\sin(\lambda RL)\cosh(\lambda SL)] \tag{3.25}$$

$$b = 2\lambda EA[(\eta S - R)\cos(\lambda RL)\sinh(\lambda SL) \tag{3.26}$$

$$- (S + \eta R)\sin(\lambda RL)\cosh(\lambda SL)] \tag{3.27}$$

Equation (3.23) can be rewritten as follows

$$\frac{\mathsf{U}(x)}{\mathsf{F}_L} = 2\left\{ \frac{\begin{bmatrix} a\cos(\lambda Rx)\cosh(\lambda Sx) \\ -b\sin(\lambda Rx)\sinh(\lambda Sx) \end{bmatrix}}{a^2 + b^2} - j\frac{\begin{bmatrix} a\sin(\lambda Rx)\sinh(\lambda Sx) \\ +b\cos(\lambda Rx)\cosh(\lambda Sx) \end{bmatrix}}{a^2 + b^2} \right\} \tag{3.28}$$

Equation (3.28) represents the receptance, $\alpha_{x,L}$, of the system, matching the final result of Derry and Stone.

With similar steps, it is possible to derive the receptance, $\alpha_{x,0}$, of a continuous bar with distributed hysteretic damping excited by an oscillating force, F_0, at the opposite end of the bar. Substituting x into Equation (3.28) for $L-x$ yields the same result. The response $\alpha_{LL} = x_L \backslash F_L$ is shown in Fig. 3.3 for a bar of length 2 m, diameter 0.2 m, Young's modulus $E = 2.0 \times 10^{11}$ N/m^2 and $\rho = 7800.0$ kg/m^3, for different η values.
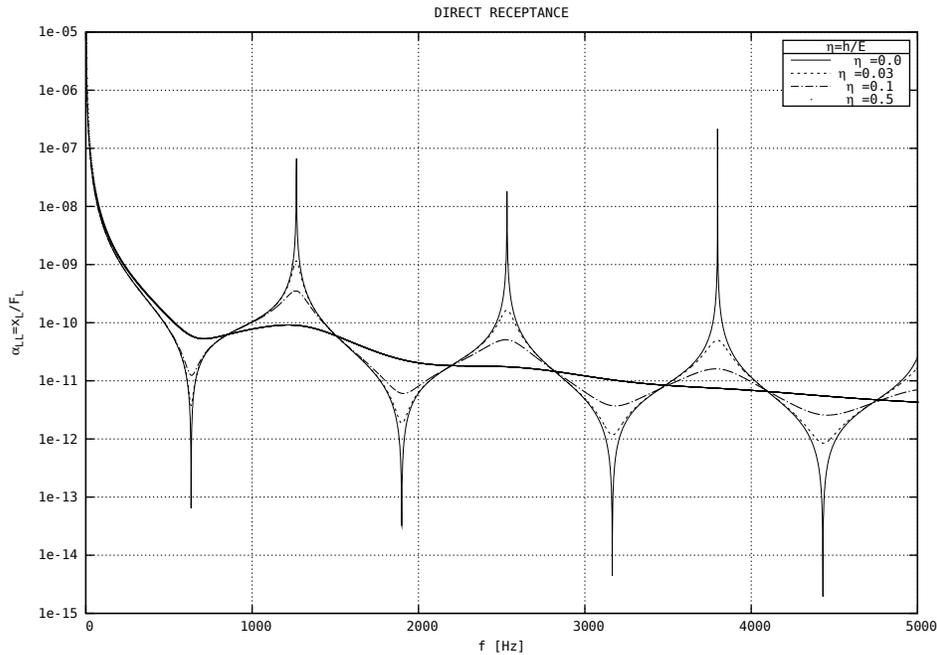
Figure 3.3: Direct receptance of a damped free/free bar for different hysteretic damping values
.
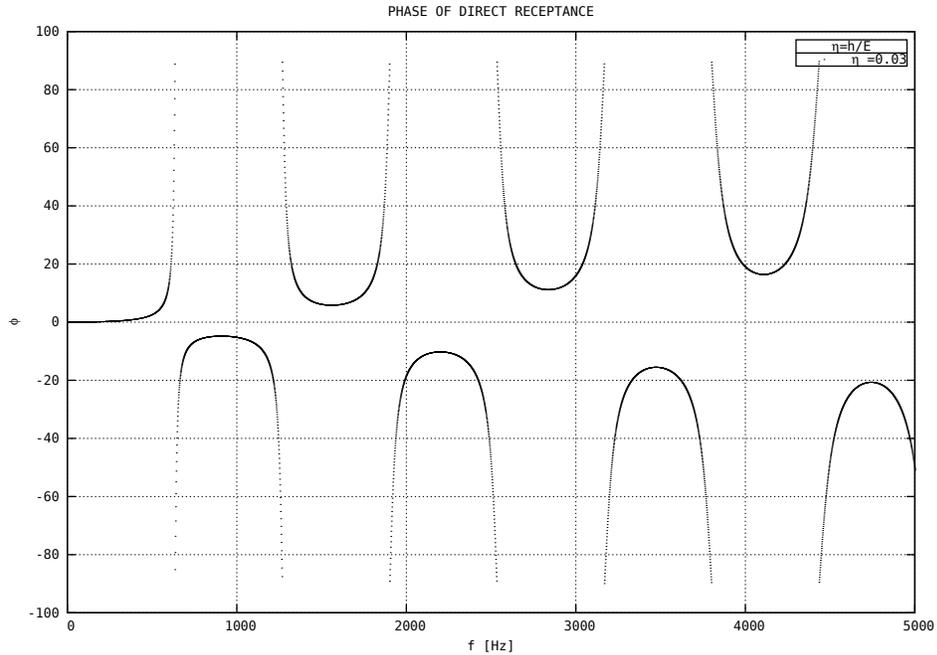
Figure 3.4: Phase of direct receptance, ϕ, for a hysteretic damping ratio of $\eta = 0.03$
.

3.3   The Universal Joint

For centuries the universal joint has been included in mechanical systems for power transmission between non-collinear shafts. This form of coupling was known long before Cardan's time $(1501 - 1576)$, who is often credited with the first description of such joints [39]. It has been reported elsewhere [45, 46] that Hooke $(1625 - 1702)$ was the first to appreciate the non uniform transmission ratio of the universal joint when constructing his 'mechanical sundial', however this type of coupling seems to have been used since Antiquity [47].

Universal joints have been used in a variety of applications, including the innovative design of a passive back-to-back torsional fatigue test rig [1] to produce an oscillating torsional moment. The arrangement consists of a motor and a pre-loaded back-to-back system incorporating a pair of gearboxes connected by a pair of double universal joint drivelines, one of which contains the rotating test specimen. The current study stems from a desire to develop a dynamic model of the entire system. The universal joint may be considered a sub-system of the back-to-back torsional fatigue test rig. As mentioned in Chapter 2, Leishman *et al.* [5, 48] and Sargeant *et al.* [20] have used receptance methods [30] to derive frequency domain models of a back-to-back gearbox rig comprising parallel shafting with different DOF. Their detailed results have confirmed that the receptance technique is well suited for the torsional modelling of back-to-back systems. In order

to model the torsional fatigue test rig, the torsional receptances of a universal joint must be known. This is the principal aim of this section.

3.3.1 Two-inertia model of a universal joint

This section presents the derivation of the torsional receptances for a universal joint comprising two inertias connected by a massless crosspiece, Fig. 3.5. Referring to Fig. 3.5, the angular positions of input axis a and output axis b are defined by ϕ_a and ϕ_b, respectively. T_a and T_b represent externally applied torques on the system. Friction forces acting on the cylindrical pairs are ignored and a constant misalignment angle θ is assumed. The kinematic relationship for a
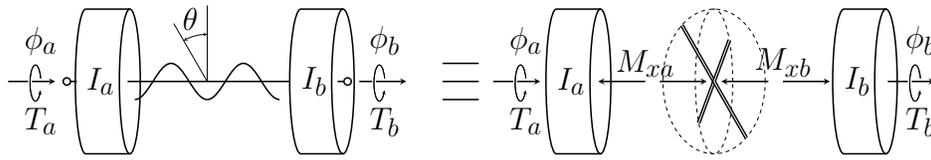
Figure 3.5: Simple model of a universal joint comprising two inertias, I_a and I_b.

universal joint has been reported by Porat [25] and is included here for completeness.

$$\tan \phi_a = \tan \phi_b \cos \theta \qquad (3.29)$$

The transmission ratio τ for a constant misalignment angle is then

$$\tau = \frac{\dot{\phi}_b}{\dot{\phi}_a} = \frac{\cos \theta}{1 - \sin^2 \theta \cos^2 \phi_a} = \frac{1 - \sin^2 \theta \sin^2 \phi_b}{\cos \theta} \qquad (3.30)$$

The moment transfer through a universal joint is well established in the literature [25] as

$$M_{xa} = \frac{M_x \cos \theta}{\sqrt{1 - \sin^2 \theta \cos^2 \phi_a}} \qquad \text{and} \qquad M_{xb} = M_x \sqrt{1 - \sin^2 \theta \cos^2 \phi_a} \qquad (3.31)$$

where M_x is the moment acting at the crosspiece. Then, according to Newton's law, the dynamic equations for inertias I_a and I_b may be written as

$$\begin{cases} I_a \ddot{\phi}_a = -M_{xa} + T_a \\ I_b \ddot{\phi}_b = +M_{xb} + T_b \end{cases} \qquad (3.32)$$

Substituting for M_{xa} and M_{xb} from Equations (3.31) and rearranging gives

$$I_b \ddot{\phi}_b = \left[(T_a - I_a \ddot{\phi}_a) \frac{1}{\tau} \right] + T_b \tag{3.33}$$

The angular acceleration $\ddot{\phi}_a$ may be expressed in terms of ϕ_b as follows

$$\dot{\phi}_a = \frac{1}{\tau} \cdot \dot{\phi}_b \qquad \text{and} \qquad \ddot{\phi}_a = -\frac{\dot{\tau}}{\tau^2} \dot{\phi}_b + \frac{1}{\tau} \ddot{\phi}_b = -\frac{1}{\tau^2} \frac{d\tau}{d\phi_b} \cdot \dot{\phi}_b^2 + \frac{1}{\tau} \cdot \ddot{\phi}_b, \tag{3.34}$$

Equation (3.33) may then be rewritten as

$$\left(\frac{I_a}{\tau^2} + I_b \right) \ddot{\phi}_b - \frac{I_a}{\tau^3} \cdot \frac{d\tau}{d\phi_b} \cdot \dot{\phi}_b^2 = \frac{T_a}{\tau} + T_b \tag{3.35}$$

The equivalent inertia of the joint measured with respect to output axis b is defined in the first term of Equation (3.35).

$$I_{rb} = \frac{I_a}{\tau^2} + I_b \tag{3.36}$$

Differentiating I_{rb} with respect to ϕ_b

$$I'_{rb} = \frac{dI_{rb}}{d\phi_b} = -2 \frac{I_a}{\tau^3} \frac{d\tau}{d\phi_b} \tag{3.37}$$

Thus Equation (3.35) reduces to

$$I_{rb} \ddot{\phi}_b + \frac{1}{2} I'_{rb} \cdot \dot{\phi}_b^2 = \frac{T_a}{\tau} + T_b \tag{3.38}$$

Equation (3.38) is the general equation of motion resolved to axis b for the simplified model of a universal joint. The equation of motion resolved to axis a may also be determined using similar steps and is given by

$$I_{ra} \ddot{\phi}_a + \frac{1}{2} I'_{ra} \cdot \dot{\phi}_a^2 = T_a + \tau T_b \tag{3.39}$$

where

$$I_{ra} = \tau^2 I_{rb} \qquad \text{and} \qquad I'_{ra} = \frac{dI_{ra}}{d\phi_a} = 2\tau I_b \frac{d\tau}{d\phi_a} \tag{3.40}$$

3.3.1.1   Receptances of a two-inertia universal joint

As noted in Chapter 2, the receptance method is well established [30] and allows steady-state models of complex systems to be constructed in the frequency domain using receptance models of sub-system components. The torsional receptance is then defined as the ratio of the displacement to the torque.

In order to derive the receptances of a universal joint, let the joint oscillate about some mean angular position at angular frequency ω. Then

$$\phi_a = \overline{\phi}_a + \tilde{\phi}_a e^{j\omega t}; \qquad\qquad\qquad \phi_b = \overline{\phi}_b + \tilde{\phi}_b e^{j\omega t} \tag{3.41}$$

$$\dot{\phi}_a = j\omega \tilde{\phi}_a e^{j\omega t}; \qquad\qquad\qquad \dot{\phi}_b = j\omega \tilde{\phi}_b e^{j\omega t} \tag{3.42}$$

$$\ddot{\phi}_a = -\omega^2 \tilde{\phi}_a e^{j\omega t}; \qquad\qquad\qquad \ddot{\phi}_b = -\omega^2 \tilde{\phi}_b e^{j\omega t} \tag{3.43}$$

For very small vibration amplitudes, $\tilde{\phi}_a$ and $\tilde{\phi}_b$, the cosine and sine functions of ϕ_a and ϕ_b can be approximated using the mean angular positions. Therefore Equation (3.30) can assume the form

$$\overline{\tau} = \frac{\cos\theta}{1 - \sin^2\theta \cos^2 \overline{\phi}_a} = \frac{1 - \sin^2\theta \sin^2 \overline{\phi}_b}{\cos\theta} \tag{3.44}$$

Similarly, Equations (3.36) and (3.40) are approximated by

$$\overline{I}_{ra} = I_a + \overline{\tau}^2 I_b \qquad \text{and} \qquad \overline{I}_{rb} = \frac{I_a}{\overline{\tau}^2} + I_b \tag{3.45}$$

Substituting Equations (3.41) to (3.45) into Equations (3.38) and (3.39) and rearranging gives

$$-\overline{I}_{rb}\omega^2 \tilde{\phi}_b e^{j\omega t} - \frac{1}{2}\left(\frac{dI_{rb}}{d\phi_b}\bigg|_{\overline{\phi}_b}\right)\omega^2 \tilde{\phi}_b^2 e^{j2\omega t} = \frac{T_a}{\overline{\tau}} + T_b \tag{3.46}$$

$$-\overline{I}_{ra}\omega^2 \tilde{\phi}_a e^{j\omega t} - \frac{1}{2}\left(\frac{dI_{ra}}{d\phi_a}\bigg|_{\overline{\phi}_a}\right)\omega^2 \tilde{\phi}_a^2 e^{j2\omega t} = T_a + \overline{\tau} T_b \tag{3.47}$$

By setting $T_a = 0$ ($T_b = 0$) in Equation (3.46), it is possible to determine the torque T_b (T_a) required to produce the oscillation $\tilde{\phi}_b$ at mean position $\overline{\phi}_b$. Again assuming very small vibration amplitudes, such that $\tilde{\phi}_b^2 << \tilde{\phi}_b$, the required torques may be approximated as

$$T_b \approx (-\overline{I}_{rb}\omega^2)\tilde{\phi}_b e^{j\omega t} = \tilde{T}_b e^{j\omega t} \qquad\qquad T_a \approx \overline{\tau}(-\overline{I}_{rb}\omega^2)\tilde{\phi}_b e^{j\omega t} = \tilde{T}_a e^{j\omega t} \tag{3.48}$$

The definition of a receptance may now be applied and α_{bb} and α_{ba} become

$$\alpha_{bb} = \frac{1}{-\omega^2 \overline{I}_{rb}} \qquad\qquad \alpha_{ba} = \frac{1}{-\omega^2 \overline{\tau} \overline{I}_{rb}} \qquad\qquad (3.49)$$

Treating Equation (3.47) in the same way, receptances α_{aa} and α_{ab} of the universal joint may also be found.

$$\alpha_{aa} = \frac{1}{-\omega^2 \overline{I}_{ra}} = \frac{1}{-\omega^2 \overline{\tau}^2 \overline{I}_{rb}} \qquad\qquad \alpha_{ab} = \frac{\overline{\tau}}{-\omega^2 \overline{I}_{ra}} = \frac{1}{-\omega^2 \overline{\tau} \overline{I}_{rb}} \qquad\qquad (3.50)$$

Consistent with Maxwell's reciprocal theorem, the equality between the cross receptances α_{ba} and α_{ab} is verified by considering Equations (3.40).

The receptances derived for the simplified model of a universal joint form part of the general case presented by Bishop and Johnson [30] for two sub-systems linked by a ratio n. In particular, the receptances given in Equations (3.49) and (3.50) have the same form as those for a gear box modelled by two inertias. There are however some important differences: for a simple gear-train, the ratio n is constant and a change in the direction of rotation occurs; for a universal joint, the ratio τ is variable and there is no change in direction of rotation. With the receptances for a two-inertia model derived it is now possible to explore its inertia variation and infer vibratory behaviour.

3.3.1.2 Inertia variation

The torsional receptance functions may be expressed in terms of the transmission ratio τ and I_{rb}, the equivalent inertia of the joint measured with respect to output axis b. Referring to Equation (3.36), I_{rb} is also a function of τ.

To investigate aspects of the universal joint's variable inertia function, it is assumed here that the joint is symmetrical with $I_a = I_b = 1$ kgm^2. Fig. 3.6a shows I_{rb}, as expressed by Equation (3.36), over one revolution of ϕ_b for different misalignment angles θ. The inertia variation of a single engine slider-crank mechanism can be explained by considering the changing geometry of the system with rotation [49]; this is also possible for the universal joint by accounting for the acceleration of the resolved element (in this case I_a). For a straight line configuration both inertias experience the same acceleration and hence the system inertia is simply the sum. For non zero θ the resolved inertia experiences a different acceleration. This arises due to the crosspiece's varying inclination with ϕ_b and results in non-linear transmittal of the contact forces which generate the moments. Regions near $90°$ ($270°$) see an increase in system inertia because in
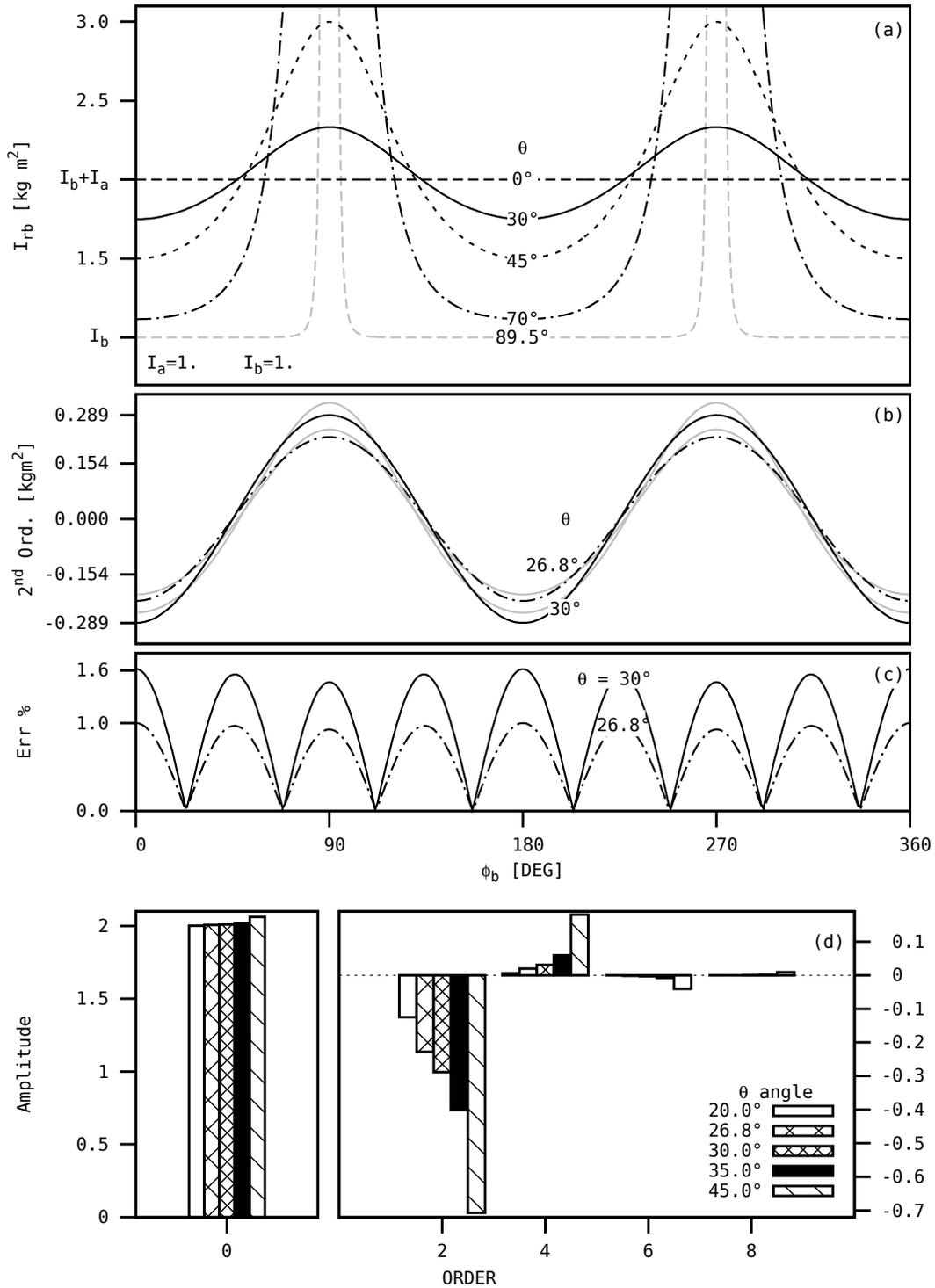
Figure 3.6: System inertia resolved to axis b (a); Comparison between the trends of I_{rb} '━━' and its second order Fourier cosine series approximation '━━' (b). The details of two values of θ are shown: $30°$ and $26.8°$; Absolute error percentage of the second order approximation (c); The Fourier Series even orders for the different misalignment angles given (d).

these positions, I_a's acceleration is greater than I_b's. Regions near $0°$ $(360°)$ and $180°$ see a reduction due to I_a's acceleration being less than I_b's. This is because, theoretically, rotation

of I_b in these locations results in rotation of the crosspiece only, and no forces and hence no moments transmitted to I_a. The variable component associated with I_a increases as θ increases. Its contribution becomes most significant at $\theta = 90°$. At this angle, when ϕ_b is near $90°$ $(270°)$, the inertia contribution tends to infinity.

Equation (3.36) is a periodic function and can be approximated by a Fourier series. It can be seen from Fig. 3.6a that, in general, for non zero misalignment angles typical of application (i.e $\leq 30°$), I_{rb} displays similar attributes to a second order cosine. It is interesting to note that this is similar to that for a single cylinder engine [50, 51]. Fig. 3.6b displays I_{rb} decreased by the linear offset $I_b + {}^1\!/_2 \cdot I_a a_0$ for two values of θ and their second order approximations. The values are $\theta = 26.8°$ and $\theta = 30°$. The former was chosen so as to result in an absolute value error $\leq 1\%$ between the cosine approximation and the full expression; the latter is representative of typical application. The absolute value errors for each case are included in Fig. 3.6c. It is also noted that the offset of system inertia is not the sum of two inertias I_a and I_b, but depends on the misalignment angle θ. In fact as θ increases the shape of the I_{rb} curve changes not only in amplitude, but so too in form. This is because the function is not simple harmonic. Due to this shape change more even order cosine components are needed to fully describe the inertia fluctuation as shown in Fig. 3.6d. Although higher order terms do contribute, it can be seen that for practical misalignment angles, below $\theta = 45°$, the variation is dominated by the second order cosine component. The presence of the frequency components will likely contribute significantly to the non-linear torsional vibration behaviour of systems incorporating these joints.

3.3.1.3   Experimental verification

It has been shown that the torsional receptances of a universal joint, derived using a simple two-inertia model, result in the joint being represented as an equivalent variable inertia I_r. In order to verify the predicted model behaviour, a test rig has been built. Figure 3.7 schematically depicts the arrangement and also gives the full list of components used. The setup mainly consists of a servomotor 1, controlled by a dynamic signal analyser B via a servodriver A1, a strain-gauged shaft, 3, and a system that integrates a standard universal joint, 8. With a selected source type B1 (swept sine, fixed sine, etc ...), the servomotor can torsionally excite the system at a given angular position (as well as at a given angular velocity, though such tests are not reported here). The average test angle ϕ_{ai} is set by a degree-wheel (error $\pm 1°$) and maintained by a system position controller P [52]. The normal moments generated by the universal joint are balanced by two bearings. A torsionally rigid coupling which is flexible in bending, 5, permits slight misalignment resulting from incorrect motor-system alignment while a pair of eccentric
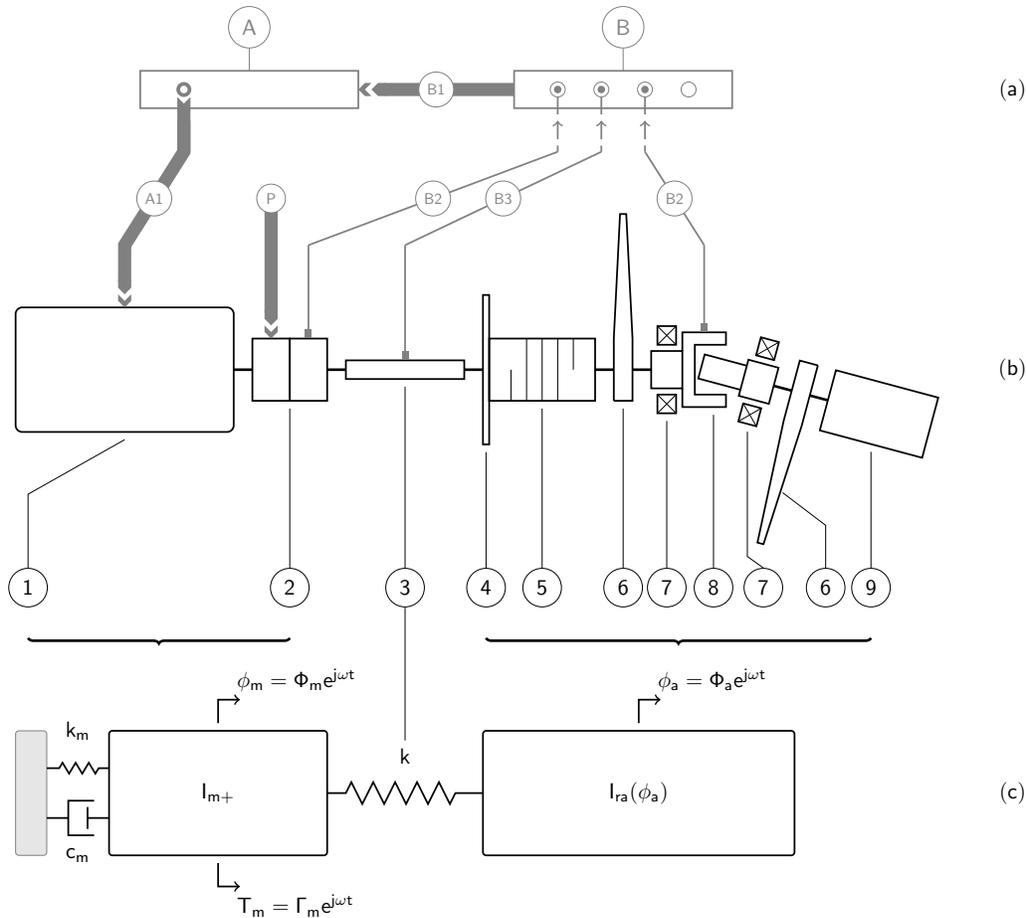
Figure 3.7: Control system, measuring chain and signal analyser: A - servodriver, A1 - servomotor control, P - telemetry system position control, B - dynamic signal analyser, B1 - excitation source type, B2 - TAP$^{\text{TM}}$ (angular) accelerometer, B3 - telemetry system torque measurement '━━' (a); test rig schematic: 1 - servomotor, 2 - coupling, 3 - shaft, 4 - degree-wheel, 5 - flexible coupling, 6 - locking plier, 7 - shaft & bearing, 8 - universal joint, 9 - inertia '━━' (b); lumped-mass system rig model (c).

elements (locking pliers), 6, asymmetrically mounted with respect to the universal joint and at appropriate angular positions, produce a localised torsional moment which acts across the joint. This appears to be the first time a locked in torque has been applied across a joint (i.e. to an open system) without the use of a back-to-back system. Such an internal preloading allows backlash effects in the cylindrical pairs of the universal joint to be removed and should enhance the torsional vibration transmission to the inclined part. Thus making the experiment more representative of the model. On the other hand, due to the increased contact forces within the pairs, it also increases frictional forces rendering the related model assumption debatable. Even so, the addition of elements, 6, make the massless crosspiece hypothesis stronger since each element serves to increase the inertia of I_a and I_b. The rig permits several misalignment angle test layouts (tested angle $\theta = 30°$; other options $\theta = 15°$, $20°$, $40°$ and $45°$). Finally, the

dynamic signal analyser converts the input signals from the TAP^{TM} (angular) accelerometers B2 and the strain-gauge B3, to angular displacement responses ϕ of inertias 2 and 8, and the excitation torque of the servomotor T_3, respectively.

The arrangement was modelled as a two degree-of-freedom (2DOF) lumped-mass system as shown in Fig. 3.7c. Conforming to the scheme of Drew *et al.* [3], the servomotor and shafting were represented by their inertias I_{m+}[1] and by a spring stiffness k_m and a viscous damper c_m accounting for the servomotor electromagnetic field. The shaft behaviour 3 was approximated as a pure spring of constant stiffness k. Elements 4 to 9, including half the shaft inertia 3, form the system under investigation $I_{ra}(\phi_a)$, according to Equation(12) of [41]. It was observed from the display on the signal analyser that the experimental results were effected by a damping component; however, its influence was found to be so small that the underlying assumptions of the model were considered well-placed [49].

With a specified angle ϕ_{ai}, the sub-system, approximated by $I_{ra}(\phi_{ai})$ and k, was expected to act as an undamped vibration absorber (or detuner) for the inertia I_{m+}, undergoing an excitation T_m at the frequency

$$f_i = \frac{\omega_i}{2\pi} = \frac{1}{2\pi}\sqrt{\frac{k}{I_{ra}(\phi_{ai})}} \tag{3.51}$$

Therefore, for discrete angles over one system revolution, experiments were conducted to measure the excitation frequency, at which the angular response $\phi_2 \approx \phi_m$ would have negligible motion. A swept sine of constant amplitude source was used to excite the system. The dynamic signal analyser processed in real-time the TAP^{TM} accelerometer and strain-gauge signals to calculate the frequency response functions (FRF) $(\Phi_2/\Gamma_3)_i$ and $(\Phi_8/\Phi_2)_i$. The lumped-mass model estimates

$$\left(\frac{\Phi_2}{\Gamma_3}\right)_i \approx \left(\frac{\Phi_m}{\Gamma_m}\right)_i = \frac{k - \omega^2 I_{ra}(\phi_{ai})}{(k_m + k - \omega^2 I_{m+} + j\omega c_m)(k - \omega^2 I_{ra}(\phi_{ai})) - k^2} \tag{3.52}$$

$$\left(\frac{\Phi_{8i}}{\Phi_2}\right)_i \approx \left(\frac{\Phi_a}{\Phi_m}\right)_i = \frac{k}{k - \omega^2 I_{ra}(\phi_{ai})} \tag{3.53}$$

Table 3.2 lists the system spring, viscous damper and inertia values used for the analysis. The frequency f_i was determined as the anti-resonance value of $(\Phi_2/\Gamma_3)_i$ and confirmed by the resonance one of $(\Phi_8/\Phi_2)_i$. Equation (3.51) gives the resonance condition of the passive control sub-system subjected to the abutment vibration at angle ϕ_{ai}. For a misalignment angle $\theta = 30°$, the measured results $(\phi_a, f)_i$ were compared with the predicted curve as shown in Fig. 3.8. It can be shown that decreasing the experimental stiffness k by 1.3%, the error percentage becomes

---

[1]The shafting, symbol subscript $+$, includes the coupling inertia 2 and half the inertia of shaft 3

Table 3.2: Experimental parameter values used in the analysis. * Taken from [49]

SYMBOL	VALUE	UNIT
k_m	50	[Nm/rad]
c_m	0.3	[Nm/rad/s]
I_{m+}	$2.783\mathrm{E}-3$	[kgm^2]
k	461*	[Nm/rad]
I_3	$5.734\mathrm{E}-5$	[kgm^2]
I_{4+5}	$1.869\mathrm{E}-4$	[kgm^2]
I_{6in}	$5.031\mathrm{E}-3$	[kgm^2]
I_{6out}	$7.845\mathrm{E}-3$	[kgm^2]
I_7	$2.420\mathrm{E}-5$	[kgm^2]
$I_{8,fork}$	$2.095\mathrm{E}-5$	[kgm^2]
I_9	$2.857\mathrm{E}-4$	[kgm^2]
θ	$30.0°$	[DEG]

less than 2.2%. It is, however, likely that the differences depend more on measurement errors of
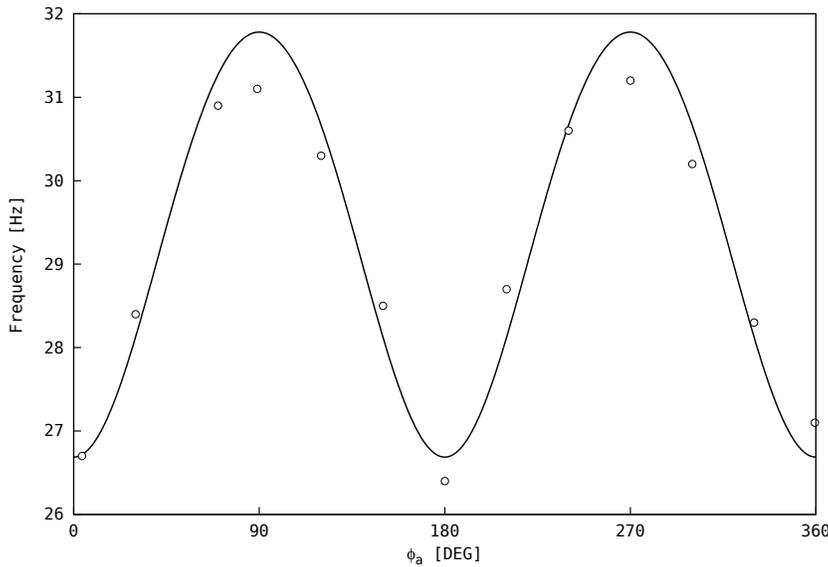
Figure 3.8: Torsional natural frequency over one revolution of a universal joint working as a detuner, Fig. 3.7c. The trends are due to the variable inertia of the universal joint, $I_{ra}(\phi_a)$: predicted curve '▬' and experimental measures 'o'.

the parameters used, on positioning of the locking pliers and on the simplistic model proposed.

As a result of theoretical and experimental frequency-based analyses, a series of rotational angles have been considered, corresponding to instantaneous snapshots representative of the underlying system behaviour. While representation of the system in this manner does not account for the non-linear effects *resulting from* inertia modulation at constant shaft velocity, the inertia variation itself; that is, the *cause* of the non-linear behaviour, can be examined. The effect of this inertia variation is illustrated in Fig. 3.8. Such an investigation could also be carried out for other varying system parameters; for example, the changing stiffness of gear systems resulting

from variable contact forces, or variable damping behaviour in other systems.

Despite its advantages, the approach undertaken produces an indirect result. In fact, based on the assumption of linear and stationary systems with constant parameters, a vibration analysis of the modelled rig in the frequency domain, in Fig. 3.7c, can show only average dynamic characteristics. Therefore, the variable inertia, $I_{ra}(\phi_a)$, has been linearised by setting Equation (3.40) parameters to

$$I_a = {}^1\!/_2 I_3 + I_{4+5} + I_{6in} + I_7 + I_{8,fork} \tag{3.54}$$

$$I_b = I_{8,fork} + I_7 + I_{6out} + I_9 \tag{3.55}$$

and the Fourier Series 0 order of τ^2 calculated for $\theta = 30°$

$$a_0\bigg|_{\theta=30°} = \frac{\cos(2\theta) + 3}{2\cos\theta}\bigg|_{\theta=30°} \tag{3.56}$$

The average value $\overline{I_{ra}(\phi_a)}$ is then expressed by

$$\overline{I_{ra}(\phi_a)} = I_a + I_b \frac{a_{0,30°}}{2} \tag{3.57}$$

Finally, simulated torsional receptances α_{mm} and α_{am} are plotted in Fig. 3.9. From which results: average natural frequencies $f_{n1} = 8.49$ Hz and $f_{n2} = 73.76$ Hz, and mean anti-resonance frequency $f = 29.35$ Hz. The experimental analysis has confirmed the dominating second order inertia variation superimposed on an average inertia. A time domain method needs to be employed to simulate the non-linear system effects due to the varying inertia.

3.3.1.4 Non-linear effects

It is well known that investigations of non-linear systems by frequency techniques yield partial results because of their implicit assumptions. In order to simulate non-linear effects of working systems, related to back-lash, friction, stiffness and inertia variations, time domain methods can overcome these limitations.

The cyclical inertia modulation of a universal joint due to the changing mechanism geometry can be processed in the same way as a reciprocating engine's apparent inertia. The effect of non-linear frequency coupling between an engine's torsional natural frequencies and mean angular speed of the engine, Ω, have been presented by Drew *et al.* [3]. Their results have been reproduced and plotted with respect to a quarter engine order increments up to the 20th engine order,
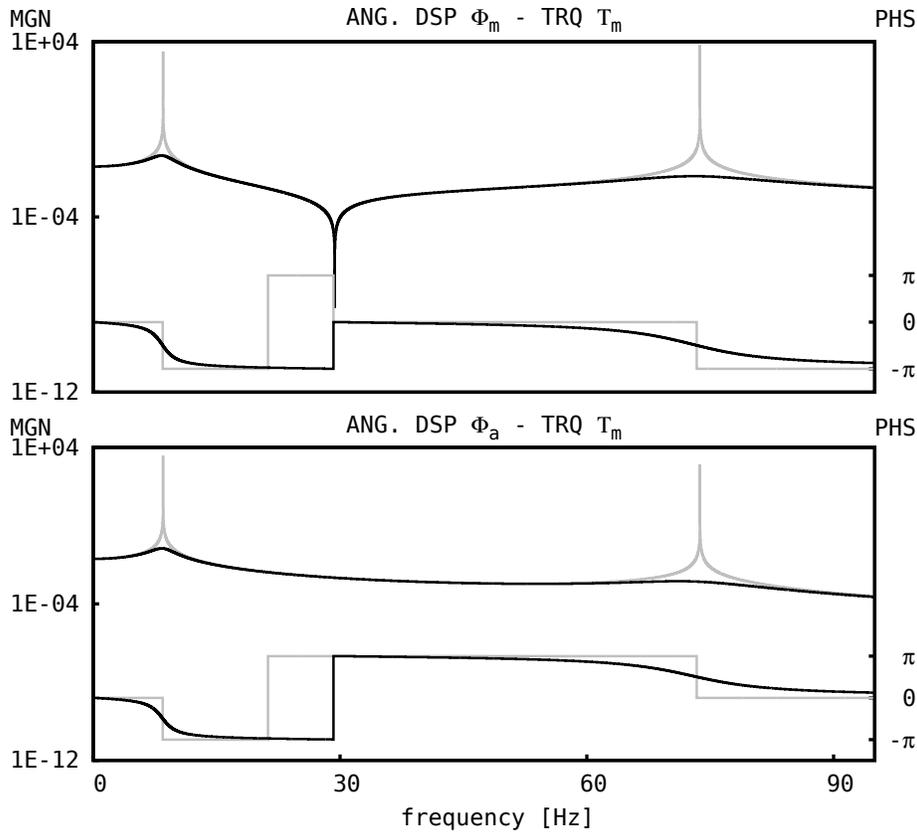
Figure 3.9: Receptances $\alpha_{m,m}$ and $\alpha_{m,a}$ of the experimental apparatus modelled as a 2DOF system. '▬' refers to simulation output with parameter values of Tab. 3.2; '▬' refers to simulation output ignoring servomotor damping

Fig. 3.10. The engine order corresponding to the frequency is defined as $n = \omega/\Omega$. It results in
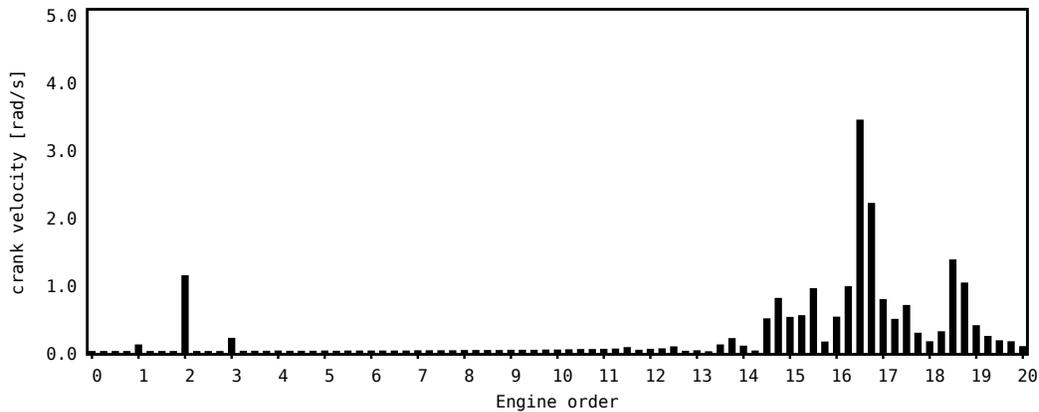
Figure 3.10: Frequency content of the crankshaft velocity by Drew's case [3].

a principal peak corresponding to the average angular natural frequency of the system, ω_n, and in main sideband structures at $\omega_n \pm 2\Omega$. The spectrum also displays smaller side-band peaks at $\omega_n \pm 1\Omega$ and $\omega_n \pm 3\Omega$.

System secondary resonances can occur in connection with excitation frequencies modulated into sidebands. Guzzomi *et al.* has then extended the investigation incorporating the effects of friction and gudgeon and/or crank pin offset [49, 50, 53].

The model scheme used by both Drew *et al.* and Guzzomi *et al.* has been conveniently modified by substituting the engine model with the two-inertia universal joint, Fig. 3.11. Furthermore, a sinusoidal excitation torque $T = 1.0\sin(2\pi f t + \phi)$ has been used to excite the universal system. The equation of motion for the servomotor is then expressed by
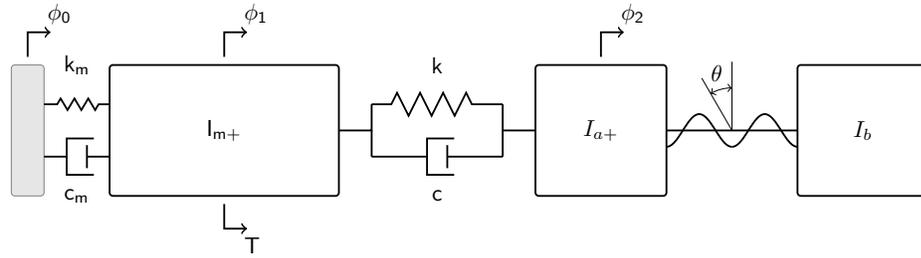
Figure 3.11: Modified scheme of Drew's rig incorporating a two-inertia universal joint for time-domain analysis.

$$I_{m+}\ddot{\phi}_1 = T - k_m(\phi_1 - \phi_0) - c_m(\dot{\phi}_1 - \dot{\phi}_0) - k(\phi_1 - \phi_2) - c(\dot{\phi}_1 - \dot{\phi}_2) \tag{3.58}$$

Using Equation (3.39) with T_b set to zero and taking T_a acting at the input co-ordinate ϕ_2

$$T_a = k(\phi_1 - \phi_2) + c(\cdot\phi_1 - \cdot\phi_2) \tag{3.59}$$

and rearranging gives

$$I_{ra}\ddot{\phi}_2 = k(\phi_1 - \phi_2) + c(\cdot\phi_1 - \cdot\phi_2) - \frac{1}{2}I'_{ra}\cdot\dot{\phi}_2^2 \tag{3.60}$$

The 4th Runge-Kutta method has been employed in solving the system consisting of Equations (3.58) and (3.60). The simulated apparatus is similar to that of the experiment, Fig. 3.7b, characterised by parameters in Tab. 3.2; however, there are some differences to note: the absence of both locking pliers, 6, which would otherwise hinder the global motion (the zero mode) and the presence of a damping in the model of shaft 3, c =0.015 [Nm/rad]. Although the non-zero damping contrasts with the assumption made in Subsection 3.3.1.3, its damping level proves to be low in torsional systems [9]. As with Drew *et al.* [3], the rig permits the universal joint sub-system to spin at an average angular velocity of = 600 rpm, (10.0 Hz,62.8 rad/s). The system

has been then excited by the superimposed sinusoidal torque T, the frequency of which has been set to the second torsional natural frequency of system, f_{n2}; two reasons support this choice. Firstly, f_{n2} is strongly affected by the driven inertia variation, I_{ra}, when all other conditions are held constant. Secondly, there is no superimposition of system responses with those of excitation frequency. A frequency-based analysis has found f_{n1} to be 19.3 Hz and f_{n2} to be 154.9 Hz for $\overline{I_{ra}(\phi_a)} = 5.95E{+}04$ kg/m^2. Upon reaching the steady-state condition, the computed angular velocity of the driven inertia has been converted from the time domain to the frequency domain using the Discrete Fourier Transform. Example frequency spectra are plotted in Fig. 3.12.
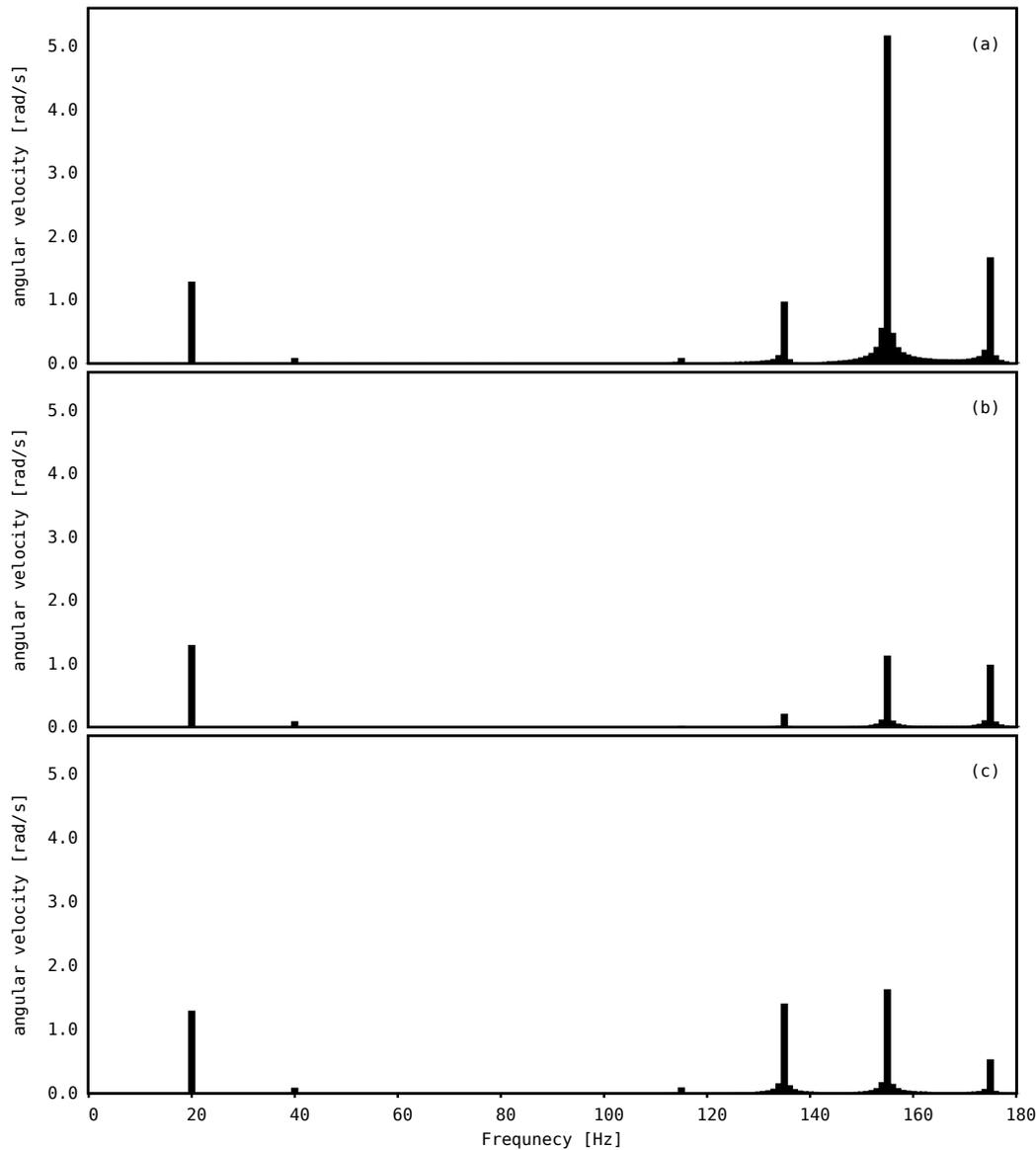
Figure 3.12: Frequency spectra of the universal joint-system angular velocity, $\dot{\phi}_2$, modulated by its inertia variation over one revolution. Average angular speed of rig 600 rpm and torsional excitation: $T = 1.0\sin(2\pi154.9t + 0.0)$ (a); $T = 1.0\sin(2\pi174.9t + 0.0)$ (b); $T = 1.0\sin(2\pi134.9t + 0.0)$ (c);

As expected, the changing inertia of the universal joint has produced a modulation of average speed, generating sideband as part of the phenomenon. The main peak corresponds to exciting torsional frequency, f, and it assumes the maximum value because of its correspondence to $f = f_{n2} = 154.9$ Hz, Fig. 3.12a. No non-linearity is linked to this frequency. According to the theory of Fourier Series, even order sidebands components appear at $\pm 2kf_\Omega$ of the mean velocity (carrier) frequency, where $k \in \mathbb{N}$ and the f_Ω is the frequency of spin rotation. The low frequency components, 20, 40 Hz.., represent the torque.

The exciting frequency of the torque T has been then modulated firstly into the upper sideband $f_{USB} = 174.9$ Hz, Fig. 3.12b, then into the lower's $f_{LSB} = 134.9$, Fig. 3.12c. Because of non-linear coupling between the low sideband and natural the frequency of the system, small sideband energy can cause big responses of the system. Consequently, operating far from the average natural frequency of the system may imply provoking secondary resonances.

In this section, the similarity between a universal joint and a single cylinder engine's inertia function has been frequently mentioned. Although there are significant differences between the two components, an investigation into the reduction of the variation in second order inertia of a single cylinder engine using a universal-joint flywheel has been investigated [54].

Drew's model has again been modified by adding a flywheel to the right extremity of the engine via a universal joint, as shown in Fig. 3.13. The rig includes two variable inertia mecha-
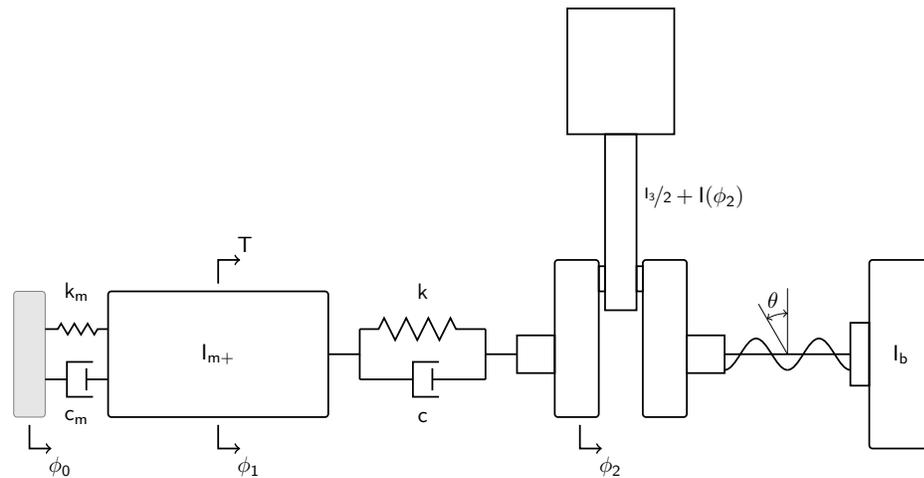
Figure 3.13: Schematic modified rig [3], incorporating a flywheel connected by a universal joint.

nisms: the engine I(ϕ_2) [3] and the universal joint-flywheel I_{ra}. The latter can be formulated by Equation (3.40) to generate a non-symmetrical universal joint $(I_a \neq I_b)$. By tuning I_a, I_b and θ, the I_{ra} inertia has been partially matched to that of the engine, Fig. 3.14a.

It has been shown that the inertia variation of a universal joint can be approximated by even
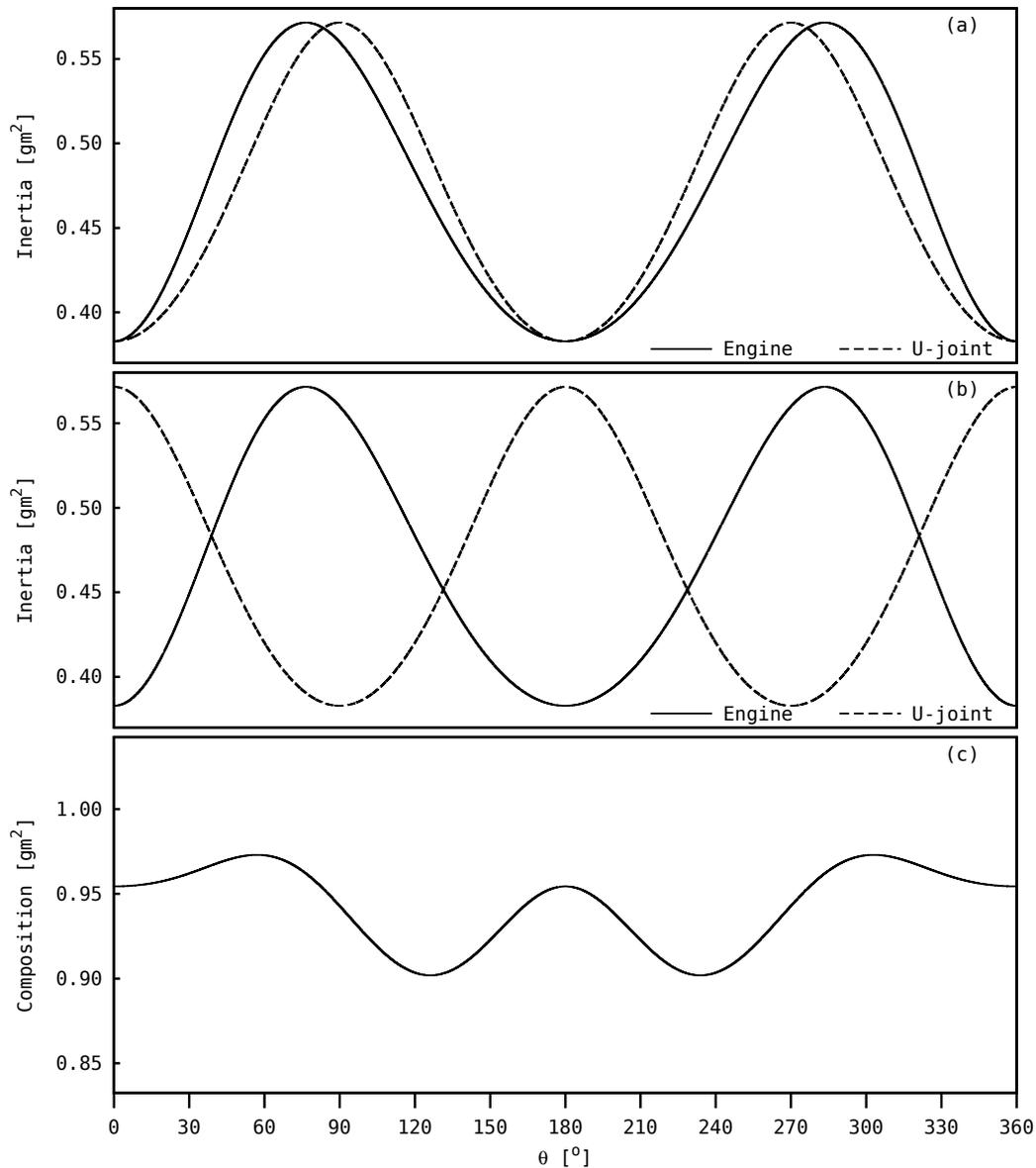
Figure 3.14: Inertias resolved to driven axis: of a single cylinder engine '━━', of a tuned universal joint '■ ■', with no phase offset (a); of a single cylinder engine '━━', of a tuned universal joint '■ ■', with 90 ° phase offset (b); of the combined systems (b)

order Fourier terms only, while the frequency content of the crankshaft velocity, Fig. 3.10, shows the presence of odd terms as well. This is the main difference. In order to reduce the inertia fluctuation of the engine, the universal joint has been rotated by 90 °, Fig. 3.14b and its inertia has been combined with that of the engine, Fig. 3.14c. The dynamic equations for the system formulating the equipment consists of Equation (3.58) and the general equation of the motion

for the combined mechanism (engine and universal joint-flywheel) as follows

$$\left[\frac{I_3}{2} + I(\phi_2) + I_{ra}\left(\phi_2 + \frac{\pi}{2}\right)\ddot{\phi}_2\right] = k(\phi_1 - \phi_2) + c(\cdot\phi_1 - \cdot\phi_2)$$
$$-\frac{1}{2}\left[I(\phi_2)' + I'_{ra}\left(\phi_2 + \frac{\pi}{2}\right)\right] \cdot \dot{\phi}_2^2 - g(\phi_2) \tag{3.61}$$

$I_3/2$ represents the inertia contribution of the shaft 3, previously included in the servomotor inertia via a subscript $+$.

In order to solve the new dynamic system, the aforementioned technique (RK4 $+$ initial conditions $\rightarrow$ standing signals $\rightarrow$ DFT) with the same average angular velocity, 600 rpm, has been used. Simulation parameters have been outlined in Tab. 3.3. For those values, the two average torsional natural frequencies of the system result: $f_{n1} = 18.3$ Hz and $f_{n2} = 127.8$ Hz. Defining the engine order as

$$n = \frac{\omega}{\Omega} \tag{3.62}$$

it results for the second average natural frequency f_{n2} in

$$n_{n2} = \frac{127.8}{10} = 12.8 \tag{3.63}$$

The second order sidebands correspond to

$$n_{n2} - 2 = 10.8 \qquad\qquad n_{n2} + 2 = 14.8 \tag{3.64}$$

In practice, a standard flywheel, characterised by a large inertia connected to the engine, is commonly employed to minimise vibration and resonance problems, in particular those of a single cylinder engine. Hence, a comparison of effects due to the standard and the tuned flywheels is appropriate. Case A refers to the rig with the standard flywheel; Case B to that with the tuned flywheel. Setting the standard flywheel inertia to the average value of the tuned flywheel implies the same average natural frequencies for both systems. In quarter engine order increments up to 20th, simulations have been conducted exciting both systems with different superimposed torques: Figure 3.15 shows the responses of both systems excited at a torque variation frequency of 127.8 Hz; Figure 3.16 shows the responses at a frequency of 147.8 Hz. Results have been subsequently compared calculating the percentage difference according to the technique developed by Guzzomi *et al.* [53]. Each frequency spectrum element of Case A is subtracted from the corresponding
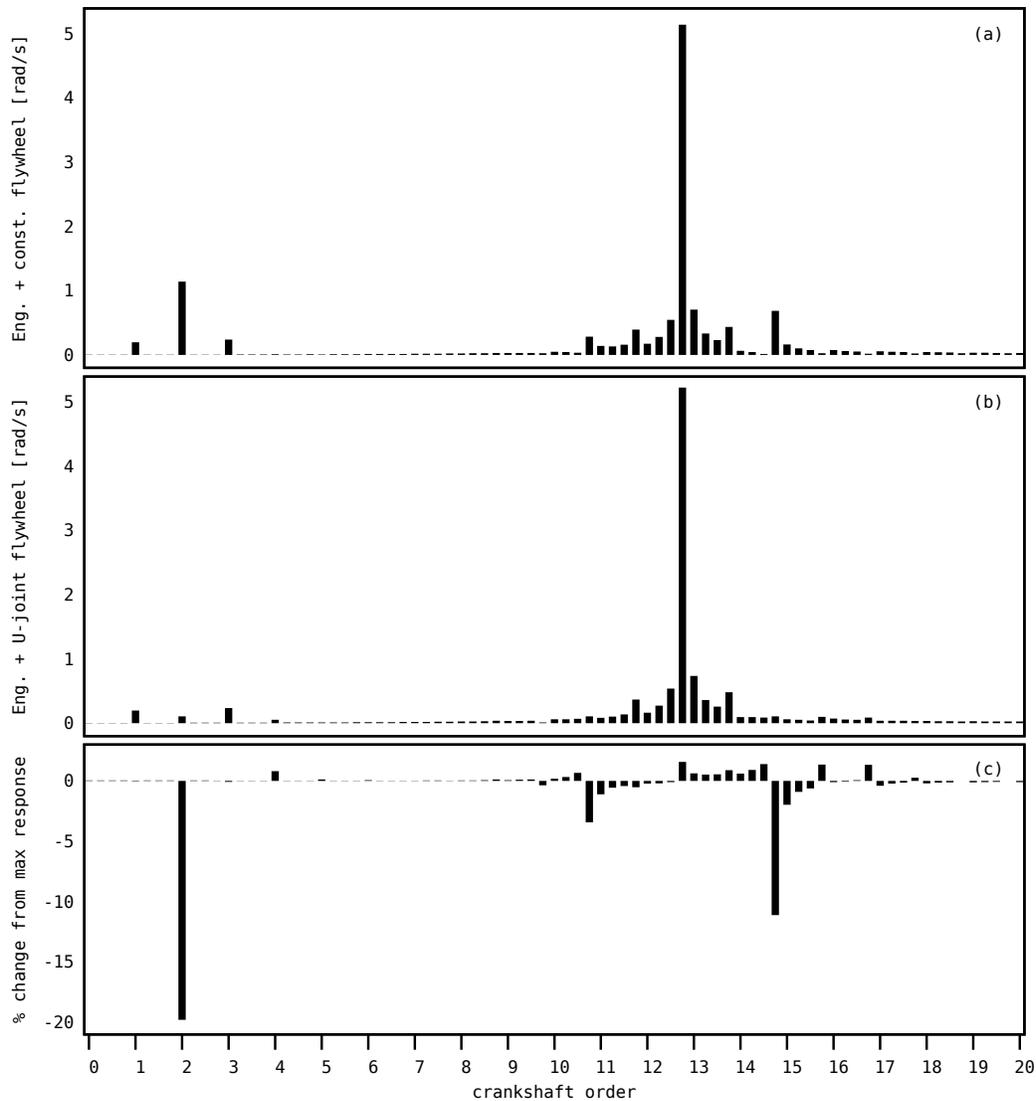
Figure 3.15: Frequency content of an engine system excited at a sinusoidal torque frequency of 127.8 Hz: with a standard flywheel - Case A (a); with a tuned universal joint flywheel - Case B (b); percentage difference between case A and Case B.

spectrum element of Case B, then these values are divided by the amplitude of response at the excitation frequency for Case A.

Figure 3.15a presents the frequency content of Case A. One notes the typical behaviour of a non-linear engine system. Comparing it to Fig. 3.10, similar remarks can be made. Figure 3.15b refers to Case B. The magnitude of the engine order corresponding to f_{n2} is not considerably changed because of its dependence on average system values; however, there is a clear reduction of the second order sideband structures due to the nature of the universal joint, confirmed in Fig.3.15c. As expected, no or negligible variations correspond to odd orders. It also results a significant reduction of the second engine order.

Again, Figure 3.15a presents the frequency content of Case A, but excited at a sinusoidal torque
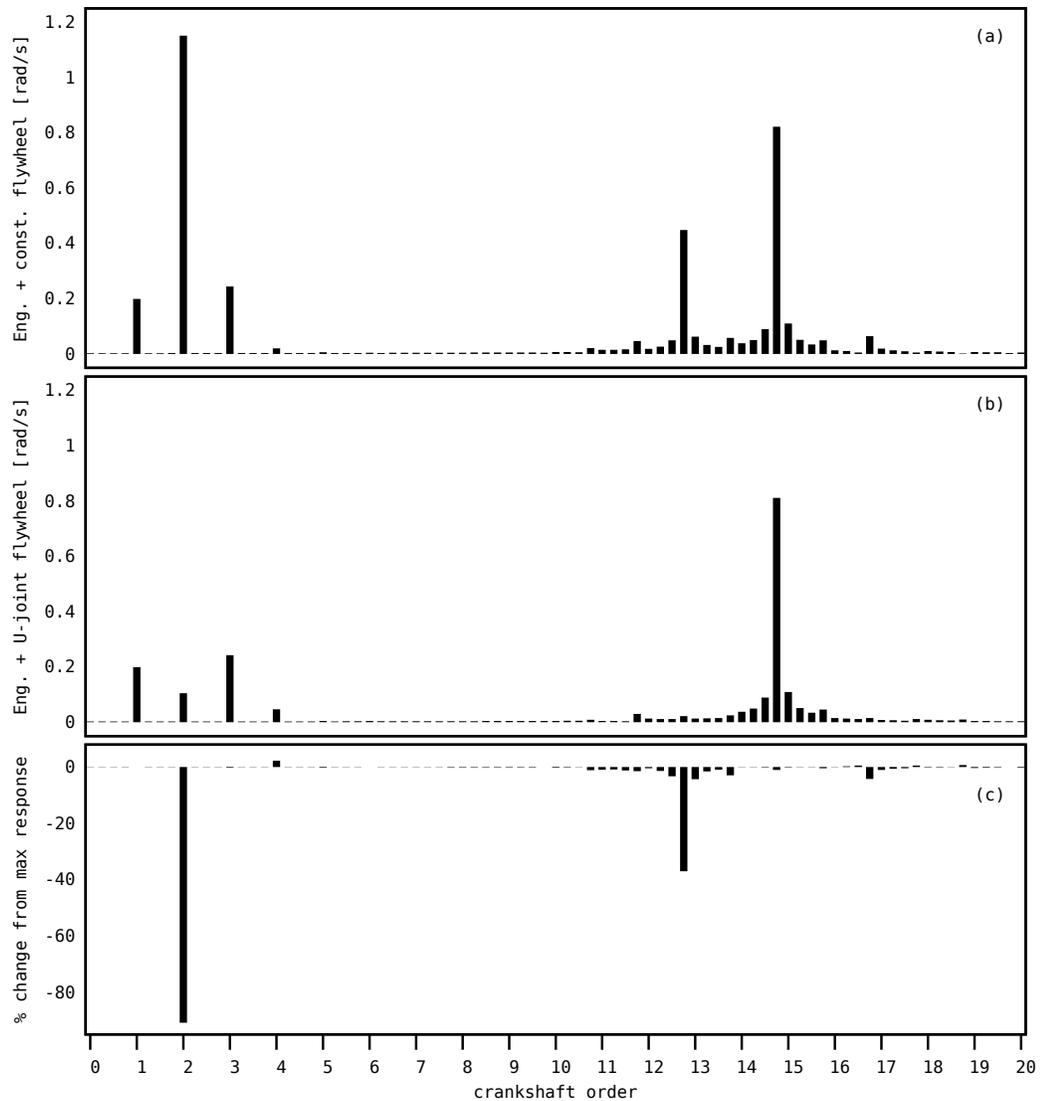
Figure 3.16: Frequency content of an engine system excited at a sinusoidal torque frequency of 147.8 Hz: with a standard flywheel - Case A (a); with a tuned universal joint flywheel - Case B (b); percentage difference between case A and Case B.

frequency of 147.8 Hz. One can note the secondary resonance phenomenon due to the interference of the low sideband with the second natural frequency of the system. Case B frequency content, fig. 3.16b displays no secondary resonance. One can appreciate the advantages of using a tuned flywheel in Fig. 3.16c.

3.3.1.5   Conclusion

Unlike existing theory on universal joints which discusses the effects of torque transmittal and velocity variations, this section has presented both a dynamic model of the joint using receptances and also investigated the attributes of the system's variable inertia. The joint was modelled as a two-inertia system. The equations of motion and resulting receptances revealed that the universal

Table 3.3: System parameter values used in the analysis.

SYMBOL	VALUE	UNIT
k_m	50	[Nm/rad]
c_m	0.3	[Nm/rad/s]
I_{m+}	$2.7830E-3$	[kgm^2]
k	461	[Nm/rad]
c	0.015	[Nm/rad/s]
I_3	$5.7340E-5$	[kgm^2]
m_p	0.2699	[kg]
m_r	0.104	[kg]
I_r	$1.5300E-4$	[kgm^2]
l	0.09847	[m]
j	0.165	[−]
m_c	0.556	[kg]
I_c	$3.2100E-3$	[kgm^2]
r	0.02491	[m]
h	0.143	[m]
I_a	0.0	[kgm^2]
I_b	$4.6780E-4$	[kgm^2]
θ	$25.22°$	[DEG]

joint may be represented as a variable inertia. The inertia variation is a function of misalignment angle and angular position. Changes in the inertia with angular position and misalignment angle were presented and discussed. The inertia variation associated with an inclined joint is not trivial. The characteristics of this variation were explained in detail. This variation may be approximated adequately, to within 1.6%, by a second order cosine for misalignment angles typical of operating conditions ($\leq 30°$).

The torsional receptances derived here prove useful for the modelling of dynamic systems incorporating such joints. As the misalignment angle increases more even order cosine terms are required to accurately describe the inertia variation. Systems with inertia variation are known to exhibit non-linear frequency coupling between rotational speed and average natural frequencies. The same is true of systems with universal joints as was demonstrated by the novel application to engine balancing.

It should however be noted that although good agreement was obtained experimentally and that the application was appropriate, the two-inertia model is limited. Both applications adopted fork inertias (I_a and/or I_b) $>>$ greater than the cross piece's and hence it was sufficient to use a model which does not include its effects. This, however, may not always be appropriate and thus the derivation of a more complete model would be useful.

3.3.2 Three-inertia model of a universal joint

In this section the dynamic study of a universal joint as a special case of a spherical four-link mechanism is presented. The input, floating, output and ground links comprise the spherical four-link mechanism. From such a mechanism a three-inertia universal joint model can be derived.

3.3.2.1 Equations of motion

For this model the dynamic equations are derived using dual algebra. The interested reader is addressed to references in the literature, e.g. [4, 55, 56, 27, 26, 57, 58, 59]. The dynamic force
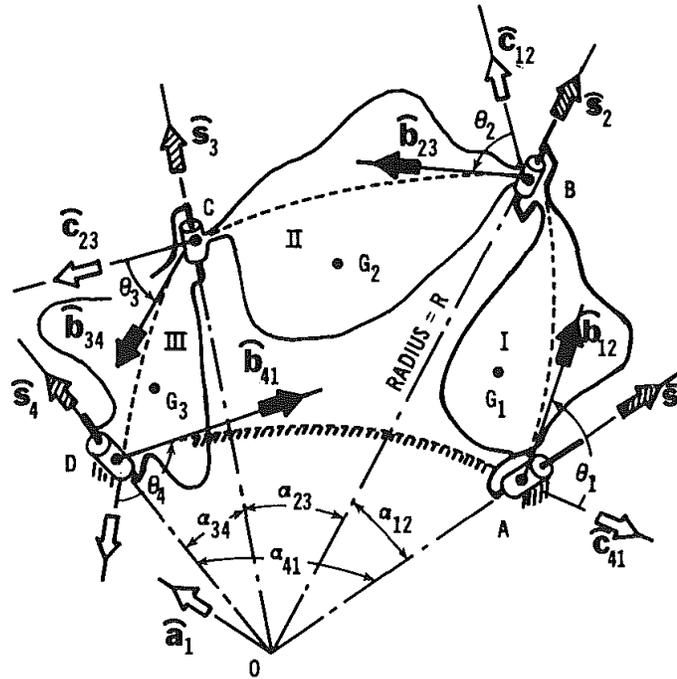
Figure 3.17: A general spherical four-link mechanism [4]. Image courtesy of ASME.

balances, as reported by Yang and Zhishang [4], are expressed by

$$\hat{\mathbf{P}}^I = [S]^T \hat{\mathbf{F}}_A^I + [\alpha_{12}]^T [S]^T \hat{\mathbf{F}}_B^I \tag{3.65}$$

$$\hat{\mathbf{P}}^{II} = [S]^T \hat{\mathbf{F}}_B^{II} + [\alpha_{23}]^T [S]^T \hat{\mathbf{F}}_C^{II} \tag{3.66}$$

$$\hat{\mathbf{P}}^{III} = [S]^T \hat{\mathbf{F}}_D^{III} + [\alpha_{34}]^T [S]^T \hat{\mathbf{F}}_C^{III} \tag{3.67}$$

The equilibrium equations of the joints B and C are formulated as follows

$$\hat{\mathbf{F}}_B^{II} + [\theta_2]\hat{\mathbf{F}}_B^I = 0 \tag{3.68}$$

$$\hat{\mathbf{F}}_C^{III} + [\theta_3]\hat{\mathbf{F}}_C^{II} = 0 \tag{3.69}$$

This study assumes their same Cartesian co-ordinate systems, their same transformation matrices and their same notations [4].

Only the dual reaction forces exerted on link III at joint D (referred to $\mathcal{F}_D^{III}$) have been modified, and an external torque T_4 added. Equation (31) of Yang and Zhishang [4] then becomes

$$\hat{\mathbf{F}}_D^{III} = \mathbf{F}_D^{III} + \epsilon\mathbf{M}_D^{III} = \begin{bmatrix} F_{D1}^{III} + \epsilon M_{D1}^{III} \\ F_{D2}^{III} + \epsilon M_{D2}^{III} \\ F_{D3}^{III} + \epsilon T_4 \end{bmatrix} \tag{3.70}$$

Rearranging Equations (3.67) and (3.69)

$$\hat{\mathbf{F}}_C^{III} = [S][\alpha_{34}]^T(\hat{\mathbf{P}}^{III} - [S]^T\hat{\mathbf{F}}_D^{III}) \tag{3.71}$$

$$\hat{\mathbf{F}}_C^{II} = -[\theta_3]^T\hat{\mathbf{F}}_C^{III} \tag{3.72}$$

and substituting $\hat{\mathbf{F}}_C^{III}$ in Equation (3.72) for Equation (3.71) gives

$$\hat{\mathbf{F}}_C^{II} = -[\theta_3]^T\big[[S][\alpha_{34}]^T(\hat{\mathbf{P}}^{III} - [S]^T\hat{\mathbf{F}}_D^{III})\big] \tag{3.73}$$

Note: S, C and Ct are used to denote sin, cos and cot functions respectively.

The dual forces acting at joint B of the floating link and at the input link, $\hat{\mathbf{F}}_B^{II}$ and $\hat{\mathbf{F}}_B^I$ respectively, can be found by using Equations (3.66), (3.73) and (3.68). Then

$$\hat{\mathbf{F}}_B^I = -[\theta_2]^T[S]\big\{\hat{\mathbf{P}}^{II} + [\alpha_{23}]^T[S]^T[\theta_3]^T\big[[S][\alpha_{34}]^T(\hat{\mathbf{P}}^{III} - [S]^T\hat{\mathbf{F}}_D^{III})\big]\big\} \tag{3.74}$$

Finally, substituting $\hat{\mathbf{F}}_B^I$ in Equation (3.65) with Equation (3.74) and rearranging yields

$$\begin{aligned} \hat{\mathbf{P}}^I &+ [\alpha_{12}]^T[S]^T[\theta_2]^T[S]\hat{\mathbf{P}}^{II} \\ &+ [\alpha_{12}]^T[S]^T[\theta_2]^T[S][\alpha_{23}]^T[S]^T[\theta_3]^T[S][\alpha_{34}]^T\hat{\mathbf{P}}^{III} \\ &= [S]^T\hat{\mathbf{F}}_A^I + [\alpha_{12}]^T[S]^T[\theta_2]^T[S][\alpha_{23}]^T[S]^T[\theta_3]^T[S][\alpha_{34}]^T[S]^T\hat{\mathbf{F}}_D^{III} \end{aligned} \tag{3.75}$$

therefore

$$\hat{\mathbf{P}}^I + [L]\hat{\mathbf{P}}^{II} + [\Lambda]\hat{\mathbf{P}}^{III} = [S]^T\hat{\mathbf{F}}^I_A + [N]\hat{\mathbf{F}}^{III}_D \tag{3.76}$$

The components of matrices $[L]$, $[\Lambda]$ and $[N]$ are listed in Appendix A.1.

Performing matrix expansion of Equation (3.76), dual-dynamic equations of a spherical four-link mechanism referenced to the co-ordinate system $\mathcal{F}^I_0$ are found and presented in compressed form

$$\hat{\mathbf{P}}^I_u + \sum_{v=1}^{3}\hat{l}_{uv}\hat{\mathbf{P}}^{II}_v + \sum_{v=1}^{3}\hat{\lambda}_{uv}\hat{\mathbf{P}}^{III}_v = \sum_{v=1}^{3}\hat{s}_{vu}\hat{\mathbf{F}}^I_{Av} + \sum_{v=1}^{3}\hat{n}_{uv}\hat{\mathbf{F}}^{III}_{Dv} \tag{3.77}$$

A universal joint with manufacturing tolerances is a spatial four-link mechanism [26, 57]. However, here these fits have not been investigated. Thus, in the absence of backlash and manufacturing tolerances, all cylindrical joint axes converge to point O and all cylindrical joint distances to O are equal. Under such conditions the spatial linkage collapses into a spherical four-link mechanism with the sphere's center at O.

As the present investigation focuses on torsional vibrations between the input and the output link of a spherical mechanism, consider the dual part of the Equation (3.77) for $u = 3$.

$$\begin{aligned}
&\hat{P}^I_3 - S\alpha_{12}(C\theta_2\hat{P}^{II}_1 - S\theta_2\hat{P}^{II}_2) + C\alpha_{12}\hat{P}^{II}_3 \\
&+ (\lambda_{31})_r\hat{P}^{III}_1 + (\lambda_{32})_r\hat{P}^{III}_2 + (\lambda_{33})_r\hat{P}^{III}_3 \\
&= \hat{F}^I_{A3} + \hat{n}_{31}\hat{F}^{III}_{D1} + \hat{n}_{32}\hat{F}^{III}_{D2} + \hat{n}_{33}\hat{F}^{III}_{D3}
\end{aligned} \tag{3.78}$$

therefore

$$\begin{aligned}
&Q^I_3 - S\alpha_{12}(C\theta_2 Q^{II}_1 - S\theta_2 Q^{II}_2) + C\alpha_{12}Q^{II}_3 \\
&+ (\lambda_{31})_r Q^{III}_1 + (\lambda_{32})_r Q^{III}_2 + (\lambda_{33})_r Q^{III}_3 \\
&= T_1 + (n_{31})_r M^{III}_{D1} + (n_{31})_d F^{III}_{D1} \\
&+ (n_{32})_r M^{III}_{D2} + (n_{32})_d F^{III}_{D2} + (n_{33})_r T_4 + (n_{33})_d F^{III}_{D3}
\end{aligned} \tag{3.79}$$

Ignoring friction acting on cylindrical pairs or turning pairs results in $M^I_{B3} = -M^{II}_{B3} = 0$ and $M^{II}_{C3} = -M^{III}_{C3} = 0$. If adequate axial clearances are provided the axial forces should become negligible [57]. Thus $F^I_{B3} = -F^{II}_{B3} = 0$, $F^{II}_{C3} = -F^{III}_{C3} = 0$ and $F^{III}_{D3} = 0$.

Substituting internal reaction forces and moments in Equations (3.79) with

$$F_{D1}^{III} = P_1^{III} - P_3^{III} Ct\alpha_{34} \tag{3.80}$$

$$F_{D2}^{III} = P_3^{II}/(S\alpha_{23}S\theta_3) + P_2^{III} - P_3^{III}Ct\theta_3/S\alpha_{34} \tag{3.81}$$

$$M_{D1}^{III} = R\left[P_3^{II}/(S\alpha_{23}S\theta_3) + P_2^{III} - P_3^{III}Ct\theta_3/S\alpha_{34}\right]$$
$$+ Q_1^{III} - Q_3^{III}Ct\alpha_{34} + Ct\alpha_{34}T_4 \tag{3.82}$$

$$M_{D2}^{III} = R\left[-P_1^{III} + P_3^{III}Ct\alpha_{34}\right]$$
$$+ Q_2^{III} + Q_3^{II}/S\alpha_{23}S\theta_3 - Q_3^{III}Ct\theta_3/S\alpha_{34} + T_4Ct\theta_3/S\alpha_{34} \tag{3.83}$$

reported here for completeness, and rearranging terms gives

$$T_1 + T_4\left[Ct\alpha_{34}(n_{31})_r + (n_{32})_rCt\theta_3/S\alpha_{34} + (n_{33})_r\right]$$
$$= Q_3^I - S\alpha_{12}(C\theta_2 Q_1^{II} - S\theta_2 Q_2^{II})$$
$$+ Q_3^{II}\left[C\alpha_{12} - (n_{32})_r/(S\alpha_{23}S\theta_3)\right]$$
$$+ Q_3^{III}\left[Ct\alpha_{34}(n_{31})_r + (n_{32})_rCt\theta_3/S\alpha_{34} + (\lambda_{33})_r\right] \tag{3.84}$$

where some components disappear because their coefficients become zero. Using Equations (A.4) and (A.7) and developing T_4, Q_3^{II} and Q_3^{III} coefficients, Equation (3.84) can be written as follows

$$T_1 + T_4\frac{S\alpha_{12}S\theta_2}{S\alpha_{34}S\theta_3} = Q_3^I - S\alpha_{12}(Q_1^{II}C\theta_2 - Q_2^{II}S\theta_2)$$
$$- Q_3^{II}\frac{S\alpha_{12}}{S\alpha_{23}S\theta_3}(S\theta_2 C\theta_3 + C\alpha_{23}C\theta_2 S\theta_3) + Q_3^{III}\frac{S\alpha_{12}S\theta_2}{S\alpha_{34}S\theta_3} \tag{3.85}$$

The second term on the left hand side of the Equation (3.85) represents an extension of Equation (37) reported by Yang and Zhishang [4]. Equation (3.85) includes the contribution of the external force T_4 acting at III-link which is a necessary requirement for torsional receptance derivation (compare Fig. 3.5)

Finally, the use of Equations (16) to (22) [4] in Equation (3.85), and letting

$$\tau = \frac{\dot{\theta}_4}{\dot{\theta}_1} = \frac{S\alpha_{12}S\theta_2}{S\alpha_{34}S\theta_3} \tag{3.86}$$

yields

$$T_1 + \tau T_4 = \left(J_{33}^I + [J_{xx}^{II}]^T[\beta_{xx}] + 2[J_{xy}^{II}]^T[\beta_{xy}] + J_{33}^{III}\tau^2\right)\ddot{\theta}_1$$
$$+ \frac{1}{2}\left(2[J_{xx}^{II}]^T[\gamma_{xx}] + 2[J_{xy}^{II}]^T[\gamma_{xy}] + 2J_{33}^{III}\tau\frac{d\tau}{d\theta_1}\right)\dot{\theta}_1^2 \tag{3.87}$$

where

$$[J_{xx}^{II}] = \begin{bmatrix} J_{11}^{II} \\ J_{22}^{II} \\ J_{33}^{II} \end{bmatrix} \qquad\qquad [J_{xy}^{II}] = \begin{bmatrix} J_{12}^{II} \\ J_{13}^{II} \\ J_{23}^{II} \end{bmatrix} \tag{3.88}$$

$$[\beta_{xx}] = \begin{bmatrix} \beta_{11} \\ \beta_{22} \\ \beta_{33} \end{bmatrix} = \begin{bmatrix} S^2\alpha_{12}C^2\theta_2 \\ S^2\alpha_{12}S^2\theta_2 \\ \left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \tau\dfrac{S\alpha_{34}}{S\alpha_{23}}C\theta_3\right)^2 \end{bmatrix} \tag{3.89}$$

$$[\beta_{xy}] = \begin{bmatrix} \beta_{12} \\ \beta_{13} \\ \beta_{23} \end{bmatrix} = \begin{bmatrix} S^2\alpha_{12}S\theta_2C\theta_2 \\ -\left(S^2\alpha_{12}Ct\alpha_{23}C^2\theta_2 + \tau\dfrac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}C\theta_2C\theta_3\right) \\ \left(S^2\alpha_{12}Ct\alpha_{23}S\theta_2C\theta_2 + \tau\dfrac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}S\theta_2C\theta_3\right) \end{bmatrix} \tag{3.90}$$

$$[\gamma_{xx}] = \begin{bmatrix} \gamma_{11} \\ \gamma_{22} \\ \gamma_{33} \end{bmatrix} = \begin{bmatrix} -S^2\alpha_{12}S\theta_2C\theta_2\dfrac{d\theta_2}{d\theta_1} \\ S^2\alpha_{12}S\theta_2C\theta_2\dfrac{d\theta_2}{d\theta_1} \\ \left(S^2\alpha_{12}Ct\alpha_{23}C\theta_2 + \tau\dfrac{S\alpha_{34}}{S\alpha_{23}}C\theta_3\right)\cdot \\ \cdot\left(-S\alpha_{12}Ct\alpha_{23}S\theta_2\dfrac{d\theta_2}{d\theta_1}+ \right. \\ \left. +\dfrac{d\tau}{d\theta_1}\dfrac{S\alpha_{34}}{S\alpha_{23}}C\theta_3- \right. \\ \left. -\tau\dfrac{S\alpha_{34}}{S\alpha_{23}}S\theta_3\dfrac{d\theta_3}{d\theta_1}\right) \end{bmatrix} \tag{3.91}$$

$$[\gamma_{xy}] = \begin{bmatrix} \gamma_{12} \\ \gamma_{13} \\ \gamma_{23} \end{bmatrix} \tag{3.92}$$

$$\gamma_{12} = S^2\alpha_{12}(C^2\theta_2 - S^2\theta_2)\frac{d\theta_2}{d\theta_1}$$

$$\gamma_{13} = 2S^2\alpha_{12}Ct\alpha_{23}S\theta_2C\theta_2\frac{d\theta_2}{d\theta_1} - \frac{d\tau}{d\theta_1}\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}C\theta_2C\theta_3 +$$

$$+ \tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}\left(S\theta_2C\theta_3\frac{d\theta_2}{d\theta_1} + C\theta_2S\theta_3\frac{d\theta_3}{d\theta_1}\right)$$

$$\gamma_{23} = S^2\alpha_{12}Ct\alpha_{23}(C^2\theta_2 - S^2\theta_2)\frac{d\theta_2}{d\theta_1} + \frac{d\tau}{d\theta_1}\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}S\theta_2C\theta_3 +$$

$$+ \tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}\left(C\theta_2C\theta_3\frac{d\theta_2}{d\theta_1} - S\theta_2S\theta_3\frac{d\theta_3}{d\theta_1}\right)$$

The equivalent inertia of the spherical four-link mechanism measured with respect to I-link, axis 1, is defined in the first term on the right-hand of Equation (3.87). Hence,

$$\begin{aligned} I_{r1} =& J_{33}^I + [J_{xx}^{II}]^T[\beta_{xx}] + 2[J_{xy}^{II}]^T[\beta_{xy}] + J_{33}^{III}\tau^2 \\ =& J_{33}^I + J_{11}^{II}S^2\alpha_{12}C^2\theta_2 + J_{22}^{II}S^2\alpha_{12}S^2\theta_2 \\ &+ J_{33}^{II}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \tau\frac{S\alpha_{34}}{S\alpha_{23}}C\theta_3\right)^2 + 2J_{12}^{II}S^2\alpha_{12}S\theta_2C\theta_2 \\ &- 2J_{13}^{II}\left(S^2\alpha_{12}Ct\alpha_{23}C^2\theta_2 + \tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}C\theta_2C\theta_3\right) \\ &+ 2J_{23}^{II}\left(S^2\alpha_{12}Ct\alpha_{23}C\theta_2S\theta_2 + \tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}S\theta_2C\theta_3\right) \\ &+ J_{33}^{III}\tau^2 \end{aligned} \tag{3.93}$$

Differentiating Equation (3.93) with respect to θ_1

$$
\begin{aligned}
I'_{r1} = \frac{dI_{r1}}{d\theta_1} = & -2(J_{11}^{II} - J_{22}^{II})S^2\alpha_{12}S\theta_2 C\theta_2 \frac{d\theta_2}{d\theta_1} \\
& + 2J_{33}^{II}\left(S^2\alpha_{12}Ct\alpha_{23}C\theta_2 + \tau\frac{S\alpha_{34}}{S\alpha_{23}}C\theta_3\right)\left(-S\alpha_{12}Ct\alpha_{23}S\theta_2\frac{d\theta_2}{d\theta_1}\right. \\
& \left. + \frac{d\tau}{d\theta_1}\frac{S\alpha_{34}}{S\alpha_{23}}C\theta_3 - \tau\frac{S\alpha_{34}}{S\alpha_{23}}S\theta_3\frac{d\theta_3}{d\theta_1}\right) \\
& + 2J_{12}^{II}S^2\alpha_{12}(C^2\theta_2 - S^2\theta_2)\frac{d\theta_2}{d\theta_1} \\
& + 2J_{13}^{II}\left[2S^2\alpha_{12}Ct\alpha_{23}S\theta_2 C\theta_2\frac{d\theta_2}{d\theta_1} - \frac{d\tau}{d\theta_1}\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}C\theta_2 C\theta_3\right. \\
& \left. + \tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}\left(S\theta_2 C\theta_3\frac{d\theta_2}{d\theta_1} + C\theta_2 S\theta_3\frac{d\theta_3}{d\theta_1}\right)\right] \\
& + 2J_{23}^{II}\left[S^2\alpha_{12}Ct\alpha_{23}(C^2\theta_2 - S^2\theta_2)\frac{d\theta_2}{d\theta_1} + \frac{d\tau}{d\theta_1}\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}S\theta_2 C\theta_3\right. \\
& \left. + \tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}\left(C\theta_2 C\theta_3\frac{d\theta_2}{d\theta_1} - S\theta_2 S\theta_3\frac{d\theta_3}{d\theta_1}\right)\right] \\
& + 2J_{33}^{III}\tau\frac{d\tau}{d\theta_1} \\
= & \; 2[J_{xx}^{II}]^T[\gamma_{xx}] + 2[J_{xy}^{II}]^T[\gamma_{xy}] + 2J_{33}^{III}\tau\frac{d\tau}{d\theta_1} \quad (3.94)
\end{aligned}
$$

Comparing Equation (3.87) with the results of Equations (3.93) and (3.94) allows Equation (3.87) to be rewritten as follows

$$
M_{E3} + M_{D3}\tau = I_{r1}\ddot{\theta}_1 + \frac{1}{2}I'_{r1}\dot{\theta}_1^2 \quad (3.95)
$$

Equation (3.95) is the general equation of the motion resolved to axis 1 of a spherical four-link mechanism.

In order to derive Equation (3.95) the Newtonian investigation provides insight into the underlying physical phenomena, in particular into that relating to kinematic pairs B and C. Such an approach also permits the effects related to friction forces to be included and investigated; though, this was considered beyond the scope of the present study.

In Appendix A.3 Equation (3.87) is also derived using an energy approach, i.e. Euler-Lagrange. The derivation of the equations governing the motion of systems by this principle typically is easy, reliable and conveniently fast. On the other hand, in a way, the physics of the problem may be lost because a global scalar quantity, i.e. the system energy, is of interest, and not what happens on a local level within the system.

The equation of motion resolved to III-link, axis 4, can be determined using similar steps or applying the transmission ratio τ defined in Equation (3.86). In fact, replacing the acceleration

and velocity components in Equation (3.95) with Equation (3.86) and its derivative with respect to time

$$
\begin{aligned}
\ddot{\theta}_1 &= -\frac{\dot{\tau}}{\tau^2}\dot{\theta}_4 + \frac{1}{\tau}\ddot{\theta}_4 \\
&= -\frac{1}{\tau^2}\frac{d\tau}{d\theta_4}\cdot\dot{\theta}_4^2 + \frac{1}{\tau}\cdot\ddot{\theta}_4
\end{aligned}
\tag{3.96}
$$

and rearranging gives

$$
\frac{1}{\tau}M_{E3} + M_{D3} = I_{r4}\ddot{\theta}_4 + \frac{1}{2}I'_{r4}\dot{\theta}_4^2
\tag{3.97}
$$

where

$$
I_{r4} = \frac{1}{\tau^2}I_{r1}
\tag{3.98}
$$

$$
I'_{r4} = \frac{d}{d\theta_4}\left(\frac{1}{\tau^2}I_{r1}\right)
\tag{3.99}
$$

Refer to Appendix A.4 for the detailed steps.

Substituting the arc lengths of the moving links α_{12}, α_{23} and α_{34} in Equations (3.89) to (3.92) for right angle reduces the analysis to the case of a standard universal joint with misalignment angle $\alpha41$ and a general position of the mass centres of the links, $\mathbf{G}_k$, $k = 1, 2, 3$ [26]. In terms of investigation of a torsional system, the mass center positions $\mathbf{G}_1$ and $\mathbf{G}_3$ of links I and III respectively do not have effects (they only increase their equivalent constant fixed axis inertia consistent with the parallel axis theorem). Actually, the mass center co-ordinates of links do not directly appear in Equation (3.85); however through the tensor of inertia related to the floating link $\mathbf{G}_2$ position affects the torsional vibration response.

Diagonal tensors of inertia yield a standard (theoretical) universal joint model, as shown in Figure 3.18.

Therefore the equivalent inertias I_{r1} and I_{r4} become

$$
I_{r1} = J_{33}^I + J_{11}^{II}C^2\theta_2 + J_{22}^{II}S^2\theta_2 + J_{33}^{II}\tau^2C^2\theta_3 + J_{33}^{III}\tau^2
\tag{3.100}
$$

$$
I_{r4} = J_{33}^I\frac{1}{\tau^2} + J_{11}^{II}C^2\theta_2\frac{1}{\tau^2} + J_{22}^{II}S^2\theta_2\frac{1}{\tau^2} + J_{33}^{II}C^2\theta_3 + J_{33}^{III}
\tag{3.101}
$$
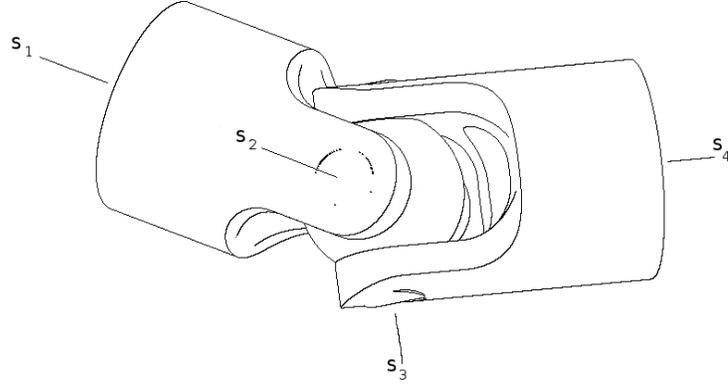
Figure 3.18: A standard universal joint with symmetrical floating link.

and I'_{r1} and I'_{r4}

$$
\begin{aligned}
I'_{r1} = &- 2(J^{II}_{11} - J^{II}_{22})S\theta_2 C\theta_2 \frac{d\theta_2}{d\theta_1} \\
&+ 2J^{II}_{33}\tau C\theta_3 \left(\frac{d\tau}{d\theta_1}C\theta_3 - \tau S\theta_3 \frac{d\theta_3}{d\theta_1} \right) \\
&+ 2J^{III}_{33}\tau \frac{d\tau}{d\theta_1}
\end{aligned}
\tag{3.102}
$$

$$
\begin{aligned}
I'_{r4} = &- 2(J^{I}_{33} + J^{II}_{22})\frac{1}{\tau^3}\frac{d\tau}{d\theta_4} \\
&- 2(J^{II}_{11} - J^{II}_{22})C\theta_2 \left(S\theta_2 \frac{d\theta_2}{d\theta_4}\frac{1}{\tau^2} + C\theta_2 \frac{1}{\tau^3}\frac{d\tau}{d\theta_4} \right) \\
&- 2J^{II}_{33}S\theta_3 C\theta_3 \frac{d\theta_3}{d\theta_4}
\end{aligned}
\tag{3.103}
$$

Equations (3.100) to (3.103) represent a compatible extension of the results reported by Peressini *et al.* [41], included in Section 3.3.1, for a universal joint modelled as two-inertias with a massless cross-piece. In fact, neglecting the floating link contribution, $J^{II}_{11} = J^{II}_{22} = J^{II}_{33} = 0.0$, of Equation (3.100) ((3.103)), the three-body model of the universal joint reduces to the simpler two-inertia one as expected, Equations (3.40) ((3.36)).

Considering Equations (7) to (10) [4], sine and cosine functions of θ_2 and θ_3 can be expressed in terms of the co-ordinate of interest, that is θ_1 for Equations (3.100) and (3.102) and θ_4 for Equations (3.101) and (3.103).

Yang [56] summarised the angular velocity ratios of a spatial four-link mechanism, i.e. in his Equations (15) to (17). Such equations, reduced to the analysed case of a standard universal joint, are the coefficients of Equation (3.100) that multiply the floating inertia J^{II}_{11} and J^{II}_{33}

respectively. The inertia J_{22}^{II} rotates with respect to an axis perpendicular to the plane defined by axes $\hat{s}_2$ and $\hat{s}_3$; however, rearranging Equation (3.100) gives

$$I_{r1} = \mathcal{I}_{33}^{I} + \mathcal{I}_{11}^{II} C^2 \theta_2 + J_{33}^{II} \tau^2 C^2 \theta_3 + J_{33}^{III} \tau^2 \tag{3.104}$$

where $\mathcal{I}_{33}^{I} = J_{33}^{I} + J_{22}^{II}$ and $\mathcal{I}_{11}^{II} = J_{11}^{II} - J_{22}^{II}$.

Referring to Equation (3.104), the equivalent inertia I_{r1} resolved to axis $\hat{s}_1$ can be now considered as the sum of a constant transformed inertia $\mathcal{I}_{33}^{I}$ rotating around axis $\hat{s}_1$, a variable transformed inertia $\mathcal{I}_{11}^{II} C^2 \theta_2$ rotating around axis $\hat{s}_2$ and the variable inertias $J_{33}^{II} \tau^2 C^2 \theta_3$ and $J_{33}^{III} \tau^2$ rotating around axes $\hat{s}_3$ and $\hat{s}_4$ respectively.

General dynamic equations resolved to any axis of joint frames $\mathcal{F}_A^{I}$ and $\mathcal{F}_D^{III}$, Fig. 3.17, can be derived from Equation (3.77). As a result, investigations on the coupled torsional and transverse vibration can be undertaken; however, transverse motion due to lateral excitation of a rotating shaft driven by a universal joint can be limited by positioning bearings close to the universal joint.

3.3.2.2   Receptances of a three-inertia model

This section shows a different derivation of torsional receptances for a sub-system, undergoing global rotation as well as torsional vibration. Let an oscillating angular displacement at coordinate p, $\theta_p e^{j\omega t}$, be specified and the oscillating torque required at coordinate q, $T_q e^{j\omega t}$, to produce this displacement be measured or modelled. Assuming that the joint spins with a constant angular velocity Ω and vibrates with a superimposed angular oscillation $\Theta e^{j\omega t}$ at angular frequency ω, then the kinematic quantities at the co-ordinate θ_1 or θ_4 can be expressed as

$$\theta_1(t) = \Omega t + \Theta_1 e^{j\omega t} \qquad\qquad \theta_4(t) = \Omega t + \Theta_4 e^{j\omega t} \tag{3.105}$$

$$\dot{\theta}_1(t) = \Omega + j\omega \Theta_1 e^{j\omega t} \qquad \text{or} \qquad \dot{\theta}_4(t) = \Omega + j\omega \Theta_4 e^{j\omega t} \tag{3.106}$$

$$\ddot{\theta}_1(t) = -\omega^2 \Theta_1 e^{j\omega t} \qquad\qquad \ddot{\theta}_4(t) = -\omega^2 \Theta_4 e^{j\omega t} \tag{3.107}$$

respectively. Substituting Equations (3.105) to (3.107) into Equations (3.95) and (3.97) gives

$$
T_1 + \tau T_4 = \frac{1}{2}\Omega^2 I'_{r1}(\theta_1)
$$
$$
+ [j\omega\Omega I'_{r1}(\theta_1) - \omega^2 I_{r1}(\theta_1)]\Theta_1 e^{j\omega t}
$$
$$
- \frac{1}{2}\omega^2 I'_{r1}(\theta_1)\Theta_1^2 e^{2j\omega t} \tag{3.108}
$$
$$
\frac{1}{\tau}T_1 + T_4 = \frac{1}{2}\Omega^2 I'_{r4}(\theta_4)
$$
$$
+ [j\omega\Omega I'_{r4}(\theta_4) - \omega^2 I_{r4}(\theta_4)]\Theta_4 e^{j\omega t}
$$
$$
- \frac{1}{2}\omega^2 I'_{r4}(\theta_4)\Theta_4^2 e^{2j\omega t} \tag{3.109}
$$

The right-hand sides of Equations (3.108) and (3.109) consist of a variable speed torque term and of first and second order oscillatory terms. The variation of the speed torque, necessary to maintain the assumption made of a constant angular velocity Ω, is due to the first derivative of inertia resolved to axes 1 and 4 with respect to θ_1 and θ_4, not to the oscillatory component of Equation (3.108).

For very small vibration amplitudes Θ_1 (Θ_4), the cosine and sine functions of $\theta_1(t)$ ($\theta_4(t)$) can be approximated using the Ωt angle, therefore

$$
T_1 + \tau(\Omega t)T_4 = \frac{1}{2}\Omega^2 I'_{r1}(\Omega t)
$$
$$
+ [j\omega\Omega I'_{r1}(\Omega t) - \omega^2 I_{r1}(\Omega t)]\Theta_1 e^{j\omega t}
$$
$$
- \frac{1}{2}\omega^2 I'_{r1}(\Omega t)\Theta_1^2 e^{2j\omega t} \tag{3.110}
$$
$$
\frac{1}{\tau(\Omega t)}T_1 + T_4 = \frac{1}{2}\Omega^2 I'_{r4}(\Omega t)
$$
$$
+ [j\omega\Omega I'_{r4}(\Omega t) - \omega^2 I_{r4}(\Omega t)]\Theta_4 e^{j\omega t}
$$
$$
- \frac{1}{2}\omega^2 I'_{r4}(\Omega t)\Theta_4^2 e^{2j\omega t} \tag{3.111}
$$

For a given time $\bar{t}$, the joint oscillates about the average angular position $\bar{\theta}_1$ ($\bar{\theta}_4$) defined by $\Omega\bar{t}$. The torque, i.e. in Equation (3.110), T_1 (T_4) for $T_4 = 0$ ($T_1 = 0$) required to produce the displacement of Equation (3.105) is modelled by a stationary torque $\overline{T}$ related to speed Ω and

an oscillatory torque $\tilde{T}$

$$T_1 = \overbrace{\frac{1}{2}\Omega^2\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right)}^{\text{stat. comp.}}$$
$$\underbrace{+\left[j\omega\Omega\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right) - \omega^2\overline{I}_{r1}\right]\Theta_1 e^{j\omega t} - \frac{1}{2}\omega^2\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right)\Theta_1^2 e^{2j\omega t}}_{\text{oscillatory comp.}} \quad (3.112)$$

$$T_4 = \overbrace{\frac{1}{2}\frac{\Omega^2}{\overline{\tau}}\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right)}^{\text{stat. comp.}}$$
$$\underbrace{+\left[j\omega\frac{\Omega}{\overline{\tau}}\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right) - \omega^2\frac{\overline{I}_{r1}}{\overline{\tau}}\right]\Theta_1 e^{j\omega t} - \frac{1}{2}\frac{\omega^2}{\overline{\tau}}\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right)\Theta_1^2 e^{2j\omega t}}_{\text{oscillatory comp.}} \quad (3.113)$$

where $\overline{\tau} = \tau(\overline{\theta}_1)$ and $\overline{I}_{r1} = I_{r1}(\overline{\theta}_1)$.

The stationary torque only shifts the mean value of oscillating angular displacement and does not influence the frequency content of the response. The vibrating torque is of interest in this investigation.

Again assuming very small vibration amplitudes, such that $\Theta_1^2 << \Theta_1$, the vibration torques may be approximated as

$$\tilde{T}_1 \approx \left[j\omega\Omega\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right) - \omega^2\overline{I}_{r1}\right]\Theta_1 e^{j\omega t} = \tilde{T}_1 e^{j\omega t} \quad (3.114)$$

$$\tilde{T}_4 \approx \left[j\omega\frac{\Omega}{\overline{\tau}}\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right) - \omega^2\frac{\overline{I}_{r1}}{\overline{\tau}}\right]\Theta_1 e^{j\omega t} = \tilde{T}_4 e^{j\omega t} \quad (3.115)$$

Applying the definition of a receptance implies

$$\alpha_{11}_{pd} = \frac{1}{j\omega\Omega\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right) - \omega^2\overline{I}_{r1}} \quad (3.116)$$

$$\alpha_{14}_{pd} = \frac{1}{j\omega\frac{\Omega}{\overline{\tau}}\left(\frac{dI_{r1}}{d\theta_1}\bigg|_{\overline{\theta}_1}\right) - \omega^2\frac{\overline{I}_{r1}}{\overline{\tau}}} \quad (3.117)$$

Equations (3.116) and (3.117) present terms depending on velocity Ω *in quadrature* (imaginary part) with the excitation as if the three-element model of a universal joint displays attributes similar to a viscous damper (i.e, $j\omega c$). Hesterman [51] was the first to recognise this pseudo-damping in the context of reciprocating engines. She noted that it would likely exist in other systems that exhibit variable inertia.

Limiting the analysis to undamped natural frequencies, only components *in phase* (real part) with the applied torque are considered, then the receptances become

$$\alpha_{11} = \frac{1}{-\omega^2 \overline{I}_{r1}} \qquad\qquad \alpha_{14} = \frac{\overline{\tau}}{-\omega^2 \overline{I}_{r1}} \qquad\qquad (3.118)$$

Treating Equation (3.111) in the same way, receptances α_{44} and α_{41} of the universal joint may also be found

$$\alpha_{44} = \frac{1}{-\omega^2 \overline{I}_{r4}} = \frac{\overline{\tau}^2}{-\omega^2 \overline{I}_{r1}} \qquad\qquad \alpha_{41} = \frac{1}{-\omega^2 \overline{\tau} \, \overline{I}_{r4}} = \frac{\overline{\tau}}{-\omega^2 \overline{I}_{r1}} \qquad (3.119)$$

Consistent with Maxwell's reciprocal theorem, the equality between the cross receptances α_{14} and α_{41} is verified by considering Equations (3.98).

The form of Equations (3.118) and (3.119) is not changed from the previous investigation [41] reported in Section 3.3.1. The universal joint is already modelled by its inertias only; however, as the equivalent inertia I_{r1} (I_{r4}) is expressed by Equation (3.100) (Equation (3.101)) it is apparent that the receptances also include contributions resulting to the floating link. It is thus now possible to investigate the effects of its inclusion through simulation.

3.3.2.3   Simulation results and discussion

The torsional receptance functions may be again expressed in terms of the transmission ratio τ and I_{r1}, the equivalent inertia of the joint measured with respect to input axis $\hat{s}_1$. Referring to Equation (3.100), I_{r1} is function of multiple transmission ratios, depending on the specific moment of inertia considered. Thus the dynamic behaviour of a universal joint is dictated largely by them. Research on other systems with variable inertia [3] and Section 3.3.1.4 have shown that such systems exhibit non-linear frequency coupling between rotation speed and the average torsional natural frequencies.

Tab. 3.4 lists the dimensions and properties of the commercial universal joint used for the analysis. The previous work on the universal joint's variable inertia function [41] and reported in Section 3.3.1 has not investigated the effect of the floating link inertia; the system was modelled as two simple inertias and the frequency analysis made for the equivalent inertia resolved to the current axis $\hat{s}_4$. It was also assumed that the joint was symmetrical with unit moments of inertia of input and output links. In Fig. 3.19a curve A proposes again the equivalent inertia function, but resolved to axis $\hat{s}_1$ for operative values of inertia moments. The misalignment angle δ is $30°$. The relative fork position between input and output link implies a phase shift of $90°$ of the

Table 3.4: Mass, center of mass and moment of inertia values for a universal joint.

Denom.	symbol	value	symbol	value	unit
		I-link (*)		*II*-link	
Mass	m^I	0.124	m^{II}	0.056	[kg]
Center of Mass	g_i^I	(0. 0. 0.025)	g_i^{II}	(0. 0. 0.)	[m]
Moment of Inertia	J_{11}^I	1.108E-4	J_{11}^{II}	3.302E-6	[kg m^2]
	J_{22}^I	1.116E-4	J_{22}^{II}	4.053E-6	[kg m^2]
	J_{33}^I	2.095E-5	J_{33}^{II}	3.302E-6	[kg m^2]

(*) For the III-link refer to I-link values and change the ^{I} with ^{III} in *symbol* box;

(◇) For co-ordinate systems refer to Yang and Zhishang [4].

(⋆) Numerical computation - open source BRL-CAD, based on geometry of actual specimen.

curve compared with the former trend. This information is also contained in the different signs of the Fourier Series even orders, but not in their absolute values, which do not vary as expected (Tab. 3.5 - OUTPUT LINK τ^2). Curve B shows I_{r1}, as expressed by Equation (3.100), over one revolution of θ_1 for the misalignment angles $\delta = 30°$. In general, for non-zero misalignment angles typical of application, I_{r1} displays similar attributes to a second order cosine.

Figure 3.19b exhibits the difference in terms of equivalent inertia between the simple two-inertia model and the three inertia, curve D, and it allows the influences of each inertia component of the floating link to be analysed qualitatively. The floating link transforms (connects) constant input physical quantities in (and) variable output ones, governed by τ. Then its properties should reflect this transition function, presenting intermediate information between input and output in terms of average and amplitude variation. Basically, this role seems to be played by the resolved inertia I_{22}^{II}, curve F. It influences on the mean value of equivalent inertia of the floating link, I_{r1}^F, and its fluctuation is in phase with the equivalent output link inertia.

The floating-link inertia variation can be also explained by considering the changing geometry of the system with rotation and, in particular, by accounting for the acceleration of the resolved elements. The kinematics of J_{11}^{II} is dictated by the input-link accelerations via the arm R, radius of the sphere. Consequently, the resolved inertia I_{11}^{II}, curve E, sees a reduction (an increase) near areas $\theta_1 = 0°$ ($90°$) and $180°$ ($270°$) since in these regions J_{33}^I's acceleration is less (greater) than J_{33}^{III}'s. In other words, for high values of the equivalent inertia, curve E, the energy has been employed to accelerate J_{11}^{II} and not to accelerate the output-link; for its small values vice versa. Similar observations can be accomplished regarding I_{33}^{II}, curve G, considering its kinematic connected to the output-link accelerations. As a result of the constraint, resolved J_{33}^{II} results in phase with the equivalent output link inertia, I_{33}^{III} and its amplitude is quite relevant to I_{r1}^F. Despite of the seemingly small inertias of the floating element for the universal joint size, Tab. 3.4, the contribution of floating element results in a system equivalent inertia increase up to ∼10%.

Table 3.5: Fourier Series even order values for different misalignment angle δ.

		Order	MISALIGNMENT ANGLE δ				
			20.0°	26.8°	30.0°	35.0°	45.0°
LINK	$C^2\theta_2$	0	0.120615	0.214828	0.267949	0.361696	0.585786
		2	−0.058432	−0.101318	−0.124356	−0.162869	−0.242641
		4	−0.001817	−0.005750	−0.008928	−0.016191	−0.041631
		6	−0.000056	−0.000326	−0.000641	−0.001610	−0.007143
		8	−0.000002	−0.000019	−0.000046	−0.000160	−0.001225
FLOATING	$S^2\theta_2$	0	1.879385	1.785172	1.732051	1.638304	1.414214
		2	0.058432	0.101318	0.124356	0.162869	0.242641
		4	0.001817	0.005750	0.008928	0.016191	0.041631
		6	0.000056	0.000326	0.000641	0.001610	0.007143
		8	0.000002	0.000019	0.000046	0.000160	0.001225
	$\tau^2 C^2\theta_3$	0	0.124485	0.227755	0.288675	0.401623	0.707107
		2	0.066053	0.126437	0.164319	0.238753	0.464466
		4	0.003987	0.013618	0.022107	0.043501	0.138564
		6	0.000184	0.001139	0.002327	0.006290	0.033875
		8	0.000008	0.000085	0.000220	0.000821	0.007545
OUTPUT LINK	τ^2	0	2.003870	2.0129263	2.020726	2.039927	2.121320
		2	0.124485	0.227755	0.288675	0.401623	0.707107
		4	0.005804	0.019369	0.031035	0.059693	0.180195
		6	0.000241	0.001465	0.002968	0.007899	0.041018
		8	0.000009	0.000104	0.000266	0.000981	0.008771

Considering Equation (3.104) and specification in Tab. 3.4, $\mathcal{I}_{11}^{II}$ can be negligible with respect to the input inertia I_{11}^{I} so that a simple three-inertia model for a universal joint can be derived only adding J_{22}^{II} to the input inertia of two-inertia model, Section 3.3.1. Figure 3.19a, Curve C, illustrates this engineering approximation. An irrational approach may consist of dividing J_{22}^{II} in equal parts and then adding one part to input inertia of two-inertia model and the other to the output one. Despite the fact that this is mathematically wrong, plotting the new model with the same parameters, Curve D does not differ too much from the mathematical formulation, Curve B.

A periodic function can be approximated by Fourier Series. Once again, the resolved inertias of a universal joint are expressed as a sum of even order cosine terms; even when all three inertias are included. More coefficient components are presented in Tab. 3.5 for some representative misalignment angles. A comparison of the floating link 0 orders can quantify the influence of J_{11}^{II}, J_{22}^{II} and J_{33}^{II} on mean value of I_{r1}^{F}. Its superior orders confirm that the oscillation amplitude is affected by J_{33}^{II} and the difference $J_{22}^{II} - J_{11}^{II}$. Fig. 3.19c shows the components of I_{r1} up until the 8th even order, using computed moments of inertia (Tab. 3.4), while Fig. 3.19d those relative to the floating link.
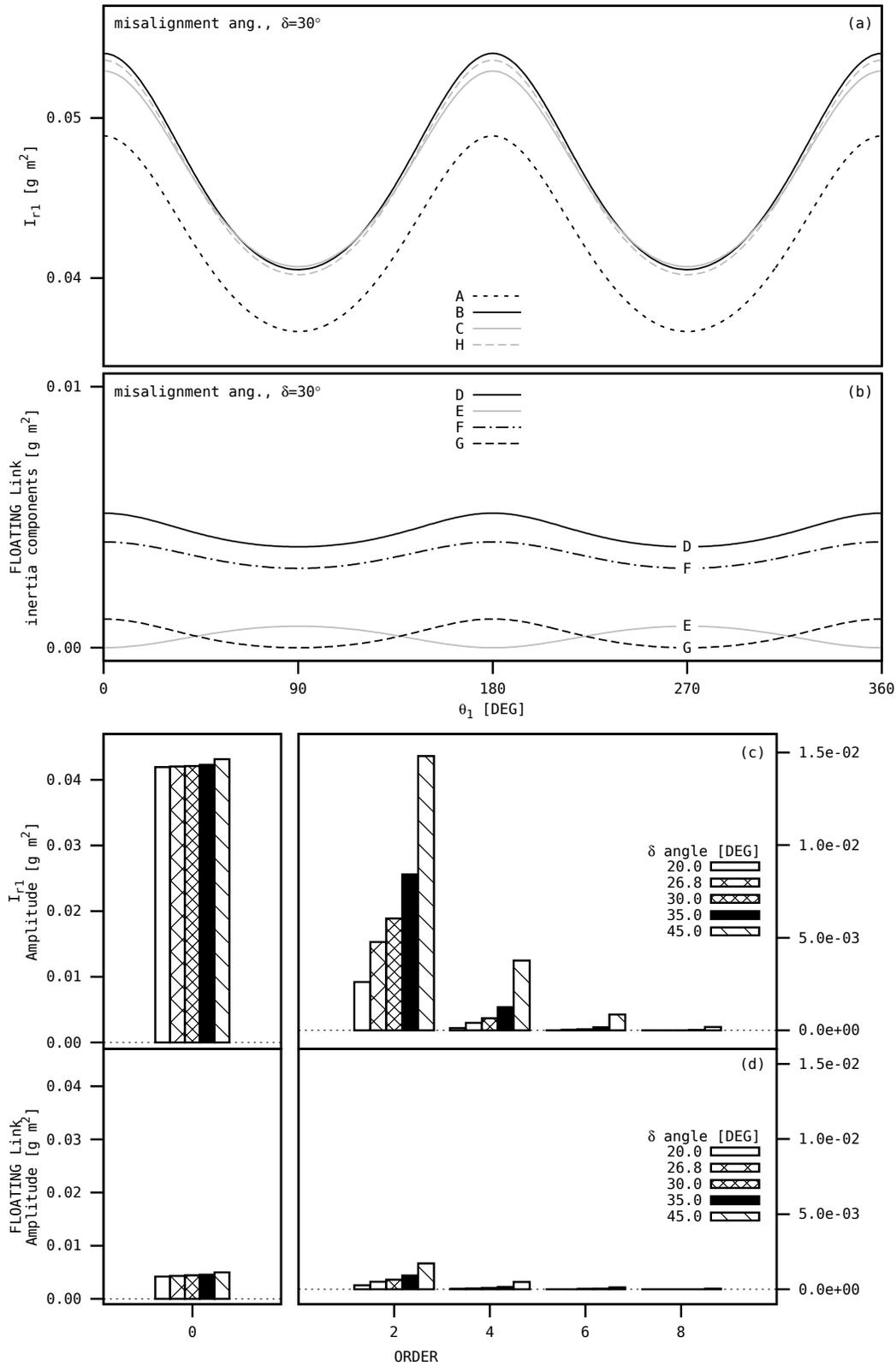
Figure 3.19: Equivalent inertia of a universal joint resolved to axis $\hat{s}_1$ for A simple two-inertia model, B four-link model and C transformed two-inertia model (a); Equivalent inertia of the floating link resolved to axis $\hat{s}_1$ trend D and its components: trend F corresponds to $J_{22}^{II}\sin^2(\theta_2)$, trend E to $J_{11}^{II}\cos^2(\theta_2)$ and trend G to $J_{33}^{II}\tau^2\cos^2(\theta_3)$ (b); Fourier series orders of I_{r1} (c); Fourier series orders of I_{r1}^F (d).

4 | Passive Torsional Fatigue test Rigs

'It is known, [...], that the repeated application and removal of a load which is considerably below the breaking weight any metallic bar will, after a number of such repeated applications, cause the fracture of the bar, and this apparent anomaly has been called the fatigue of metals [6]'. '

–

In this chapter two innovative passive torsional fatigue test rigs have been investigated and, based on the receptance technique, their dynamic models in the frequency domain have been developed. Although the rigs differ in construction, Fig. 4.1 and Fig. 4.2.1, they consist of the same components, presented in Chapter 3. Amongst these, the universal joint plays the most important role. In fact, for continuous shaft rotation, it passively causes the oscillating torque required to stress specimens. It seems that no further applications exploiting the non-linear behaviour of the universal joint have been conceived since the mechanical sundial of Hooke $(1625 - 1702)$ [46].

Transverse motion due to lateral excitation of rotating shafts connected to both a gearbox and a universal joint are present in Ref.s [20, 23, 60]. In order to limit lateral vibration, bearings are typically positioned close to the exciting source components. In the current thesis, both rigs have been constructed according to this principle so that lateral vibration have been minimised. Investigation into other effects, as well as their various combinations (e.g. variable gear mesh stiffness and/or variable universal joint inertia), have been left as future work.

It is well known that damping levels in rotating systems are normally low, leading to the possibility of failures that occur suddenly due to resonance problems [52]. Thus, predicted recep-

tances and deflected shapes of both the single and double back-to-back rigs have been derived
for mapping off their dynamic characteristics.

4.1 Single back-to-back rig

In order to perform torsional fatigue tests on splined shafts, Guzzomi and Molari [1] presented
an innovative use of a back-to-back rig, a simple scheme of which is shown in Fig. 4.1. As
noted in Chapter 1, this arrangement had been used by Fischer *et al.* [22] to experimentally
investigate the validity of assumptions regarding the influence of the intermediate-joint axial
force of a universal joint on a driveline transmitting torque at high speed. Guzzomi and Molari
independently conceived the rig for the purpose of performing rapid torsional fatigue tests with
reduced energy inputs compared to current practice (e.g. torsional hydraulic systems and motor-
driven systems with external torque control of the input and/or output [14, 15]).

Figure 4.1: For readers' convenience reproduction of test rig schematic and used notation: single back-
to-back system [1, 22].

The setup essentially consists of a servomotor and a back-to-back system. A pair of double
universal joint drivelines, (a, s) to (c, s) for $s = 1, 2$, one of which includes a specimen, (b, s), are
connected to two spur gearboxes of radius R_1 and R_2. Taking $R_1 \leq R_2$, the gear ratio, i, can
be expressed by the following relationship:

$$i = \frac{\omega_2}{\omega_1} = \frac{R_1}{R_2} = \frac{M_1}{M_2} \tag{4.1}$$

A pre-loaded moment, M_{PL}, is applied to the rig by imposing an angular displacement, θ_{PL},
resulting in both the removal of back-lash in the kinematic pairs and the application of an average

test torque. Consequently, the specimen is subject to combined steady and, with correct bearing positioning, cyclic pure torsion stresses. In the case of no pre-load, or a low value thereof, no torsional alternating shear stress results. Technically, the rig scheme of Fig. 4.1 is a 3 DOF system comprised of: an externally imposed angular displacement, θ_{PL}, a universal joint misalignment angle, δ and the gearbox ratio, i. Fixing δ and i for operative conditions, however, reduces the number of DOF to 1. The amplitude and mean value of the stress cycle cannot be independently set. This represents a considerable constraint on performing fatigue tests; however, it is possible to derive S-N curves (Wöheler curves) for different specimens with a hybrid approach; part experimental and part theoretical.

4.1.1   Static analysis

A first investigation has focused on the derivation of static torsional moments acting on the extremities of each back-to-back component over one revolution of the rig, due to the locked pre-load within. A static model has been derived as follows. Referring to Fig. 4.1, the external imposed angular displacement, θ_{PL}, is distributed to both double universal joint drivelines, θ_1 and θ_2. The equation of compatibility gives:

$$\theta_{PL} = \theta_1 + \theta_{21} \tag{4.2}$$

Where, neglecting the mesh stiffness and damping in gearboxes:

$$\theta_{21} = \frac{\theta_2}{i} \tag{4.3}$$

The angular displacements in Equations (4.2) and (4.3), θ_1 and θ_2, can then be expressed as functions of their components, shafts a,b and c, respectively, resolved to axes 1 and 2. Therefore:

$$\theta_i = \theta_{a,s} + \theta_{b,ss} + \theta_{c,s} \qquad \text{for } s = 1,2 \tag{4.4}$$

Due to the universal joint's transmission ratio of angular displacements, Equation (3.29), gives:

$$\theta_{b,ss} = \arctan(\tan\theta_{b,s} \cdot \cos\delta) \tag{4.5}$$

Included here for completeness, the transmission ratio for a misalignment angle δ is (to be compared with Equation (3.30)):

$$\tau_{pq,s} = \frac{\dot{\theta}_{q,s}}{\dot{\theta}_{p,s}} = \frac{M_{p,s}}{M_{q,s}} = \frac{\cos\delta}{1 - \sin^2\delta\cos^2\theta_{p,s}} = \frac{1 - \sin^2\delta\sin^2\theta_{q,s}}{\cos\delta} \tag{4.6}$$

Subscript p refers to the input shaft while subscript q refers to the output shaft. Rearranging Equations (4.2), (4.4) and (4.5) yields:

$$\theta_{PL} = \theta_{a,1} + \arctan(\tan\theta_{b,1} \cdot \cos\delta) + \theta_{c,1}$$
$$+ \frac{1}{i}[\theta_{a,2} + \arctan(\tan\theta_{b,2} \cdot \cos\delta) + \theta_{c,2}] \tag{4.7}$$

Considering a linear torque-angle relationship for all shafts, (i.e. $k = const.$), Equation (4.7) becomes:

$$\theta_{PL} = \frac{M_{a,1}}{K_{a,1}} + \arctan\left[\tan\left(\frac{M_{b,1}}{K_{b,1}}\right) \cdot \cos\delta\right] + \frac{M_{c,1}}{K_{c,1}}$$
$$+ \frac{1}{i}\left\{\frac{M_{a,2}}{K_{a,2}} + \arctan\left[\tan\left(\frac{M_{b,2}}{K_{b,2}}\right) \cdot \cos\delta\right] + \frac{M_{c,2}}{K_{c,2}}\right\} \tag{4.8}$$

Equation (4.8) can then be expressed as a function of $M_{a,1}$. Using Equations (4.1) and (4.6) and rearranging gives:

$$\theta_{PL} = \frac{M_{a,1}}{K_{a,1}} + \arctan\left[\tan\left(\frac{M_{a,1}}{\tau_{ab,1} \cdot K_{b,1}}\right) \cdot \cos\delta\right] + \frac{\tau_{cb,1}M_{a,1}}{\tau_{ab,1}K_{c,1}}$$
$$+ \frac{1}{i}\left\{\frac{M_{a,2}}{K_{a,2}} + \arctan\left[\tan\left(\frac{M_{a,1}}{i \cdot \tau_{ab,2} \cdot K_{b,2}}\right) \cdot \cos\delta\right] + \frac{\tau_{cb,2}M_{a,2}}{\tau_{ab,2}K_{c,2}}\right\} \tag{4.9}$$

Equation (4.9) allows torsional moment, $M_{a,1}$, to be predicted as a function of the external pre-load, θ_{PL}, over one rig revolution for different setups of universal joints and gearbox ratios. Having calculated $M_{a,1}$, it is easy to derive the other element moments using Equations (4.1) and (4.6). Correspondingly, a zero external pre-load, $\theta_{PL} = 0$, leads to $M_{a,s} = 0$ and, via Equation (4.6), $M_{b,s} = 0$. Therefore, in this condition, there is no alternating shear stress cycle. In addition, the superimposed moments resulting from the servomotor and friction, not included in the current model, would likely add such a small contribution that one can neglect their effects. Again, in this case, the added stress is likely to have a minimal non-alternating shear stress component.

Setting $\tau_{ab,1} = \tau_{cb,1} = \tau_1$, $\tau_{ab,2} = \tau_{cb,2} = \tau_2$ and taking the stiffness of shafts (a, s) and (c, s) to

be equal, the pre-loaded angular displacement can be expressed as follows:

$$\theta_{PL} = 2 \cdot \frac{M_{a,1}}{K_{a,1}} + \arctan\left[\tan\left(\frac{M_{a,1}}{\tau_1 \cdot K_{b,1}}\right) \cdot \cos\delta\right]$$
$$+ \frac{1}{i}\left\{2 \cdot \frac{M_{a,1}}{i \cdot K_{a,2}} + \arctan\left[\tan\left(\frac{M_{a,1}}{i \cdot \tau_2 \cdot K_{b,2}}\right) \cdot \cos\delta\right]\right\} \tag{4.10}$$

Using Equation (4.10), two setups have been analysed for a misalignment angle of $\delta = 30°$ with identical driveline components, such that $K_{a,1} = K_{a,2}$ and $K_{b,1} = K_{b,2}$. The first layout, indicated as Setup A, sees the two drivelines, 1 and 2, with phase angles of $0°$, resulting in $\tau_1 = \tau_2$. The second layout, indicated as Setup B, sees the two drivelines, 1 and 2, rotated at $90°$, resulting in $\tau_{1,90°} = \tau_2$. Setup B is shown in Fig. 4.1.
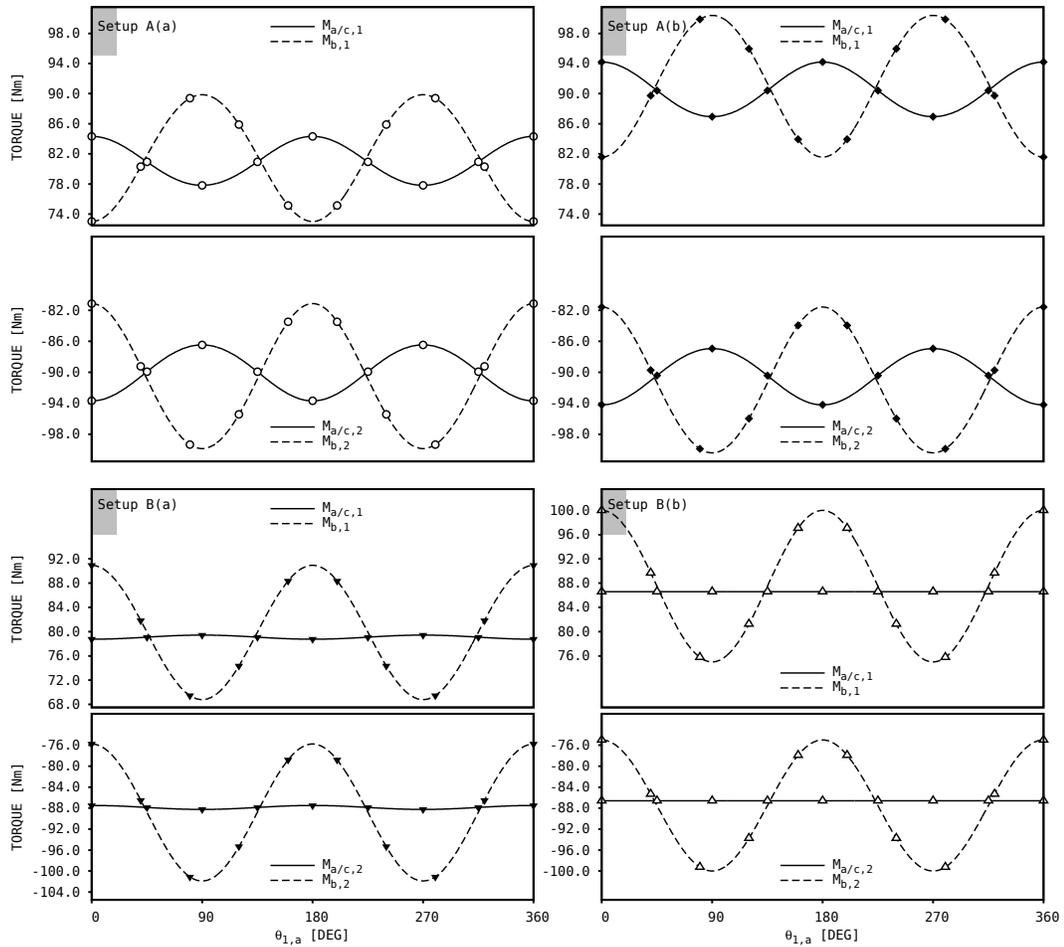
Figure 4.2: Torsional moments acting on back-to-back system components due to an external pre-load angular displacement. Setup A refers to drivelines 1 and 2 with phase $0°$; Setup B refers to drivelines 1 and 2 with phase $90°$; (a) refers to $i \neq 1$; (b) refers to $i = 1$; '▬' refers to $M_{a/c,1}$ and $M_{a/c,2}$; '■ ■' refers to $M_{b,1}$ and $M_{b,2}$.

In addition, the effects of two different gearbox ratios, i, on the moments have been investigated.

Table 4.1: Static model physical data, units S.I.

El.	Symbols	Values
δ		$\pi/6$
R_1	-	0.08573
R_2	-	0.09525
$k_{a/c,1/2}$	G ρ L d	8.E+10 7800.0 0.05 0.02
$k_{b,1/2}$	G ρ L d	8.E+10 7800.0 0.25 0.02

The notation assigns $i \neq 1$ to (a) and $i = 1$ to (b).

Results from the numerical simulations are shown in Fig. 4.2, whilst element specifications are given in tab. 4.1. According to the model, Setup A produces fluctuating torsional moments acting on the gearboxes of the same magnitude order as those acting on the inclined shafts for any i. Setup B, however, results in an important reduction in the aforementioned fluctuations, culminating in zero for $i =1$, as shown in Fig. 4.2 Setup B(b). Thus, due to conservation of energy, an amplitude increase is observed in $M_{b,s}$. A gearbox ratio different from 1 yields larger moments in the driveline. This different distribution of the internal loads would lead to a different design of the double universal joint driveline. On the other hand, a gearbox ratio of 1 produces identical amplitude and mean value moduli for the torsional moments acting on elements $\widehat{a, s}$, $\widehat{b, s}$ and $\widehat{b, s}$. To complete the investigation of this arrangement, a dynamic model in the frequency domain must be developed and analysed. According to the principle of superpositioning, significant dynamic effects could be added to the static behaviour of the system.

4.1.2   Dynamic analysis

The torsional back-to-back fatigue test rig has been modelled in the frequency domain with the receptance method, previously described in Chapter 2. Components, required to model the system, have been presented and, in several cases, developed and investigated in Chapter 3. Amongst these, three subsystems have non-linear torsional behaviour: the servomotor, the gearbox and the universal joint. Because of the inherent assumptions of the sub-structuring approach in the frequency domain, however, the systems have been linearised. Table 3.1a-c lists the models of the servomotor and gearbox [48, 52], while the universal joint, modelled as a two-inertia system, has been implemented via Equation (3.57) with I_a and I_b in Tab. 3.4 [1]. The choice of the two-inertia model was dictated by the low impact of the equivalent floating inertia , I_{r1}^F of Fig. 3.19, on the rig dynamic behaviour. The introduction of stiffness and/or damping in the

---

[1] $I_a = I_b = I_{33}^I = I_{33}^{II}$

kinematic pairs of the universal joint could imply a different selection.

The frequency-based back-to-back rig model is shown in Fig. 4.3, which also lists rig components. The multi-degree-of-freedom system is composed of a combination of lumped-mass sub-systems and continuous shafts with distributed hysteretic damping. Considering the fact that the universal joints have been modelled with their average inertias (Fourier Series 0 order), the dynamic responses of the arrangement is similar to that investigated by Leishman *et al.* [5]. Figure 4.4 shows both the magnitude and phase components of the receptances of the simple back-to-back rig for a universal joint misalignment angle of $\delta = 30°$. As the torsional fatigue test rig is currently in the design phase, the findings of Leishman *et al.* represent an important reference for comparison. By using similar values to those employed in their study for the rig component parameters[2], reported in Tab. 4.3, and by setting the universal joint misalignment angle, δ, and the relative inertias to zero, simulated frequency response functions have been derived. These figures are in good agreement with the results of Leishman *et al.* over the range 0-1600 Hz. Drew and Stone [3, 5] demonstrated a working range of their servomotor up to at least 1600 Hz. Under these conditions, torsional stiffness, k_m, and torsional damping, c_m, are expected to remain within the ranges of 40-50 [Nm/rad] and 0.2-0.4 [Nm/rad/s], respectively. Furthermore, viscous gear mesh damping, c, has been accounted for by applying the expression of Yoon and Rao [61]:

$$c = 2\xi_g \left[\frac{k}{R_0^2/I_0 + R_1^2/I_1} \right]^{\frac{1}{2}} \tag{4.11}$$

The range of the percentage critical damping for spur gear pairs, estimated by Yoon and Rao, is quiet broad, $\xi_g \in [0.03, 0.17]$. Therefore, a mean value of 0.10% has been used for the model, although the experimental results of Drew and Stone [9, 62] report a value of 0.17%. The corresponding gearbox damping ratio is $\sim$5.3 Nm/rad/s.

Experimental frequency response functions were obtained by Leishman *et al.* [5] by measuring the input excitation (servomotor input voltage) and response (measured laser torsional vibration). The servomotor excitation results in abutment vibration, corresponding to coordinate 0 of the proposed model; however, modelling the servomotor with a fixed abutment, they simulated the receptances by applying the equivalent torque at coordinate 1 of Fig. 4.3. In order to confirm the developed receptance program, listed in Appendix C.1, similar conditions have been used by the author. As a frequency-based servomotor model with abutment excitation has also been derived, Tab. 3.1, substituting such a model for that of fixed abutment, as lead to new frequency-based

---

[2]maintaining the Slave and Test gearbox specifications in Table 1 [5].
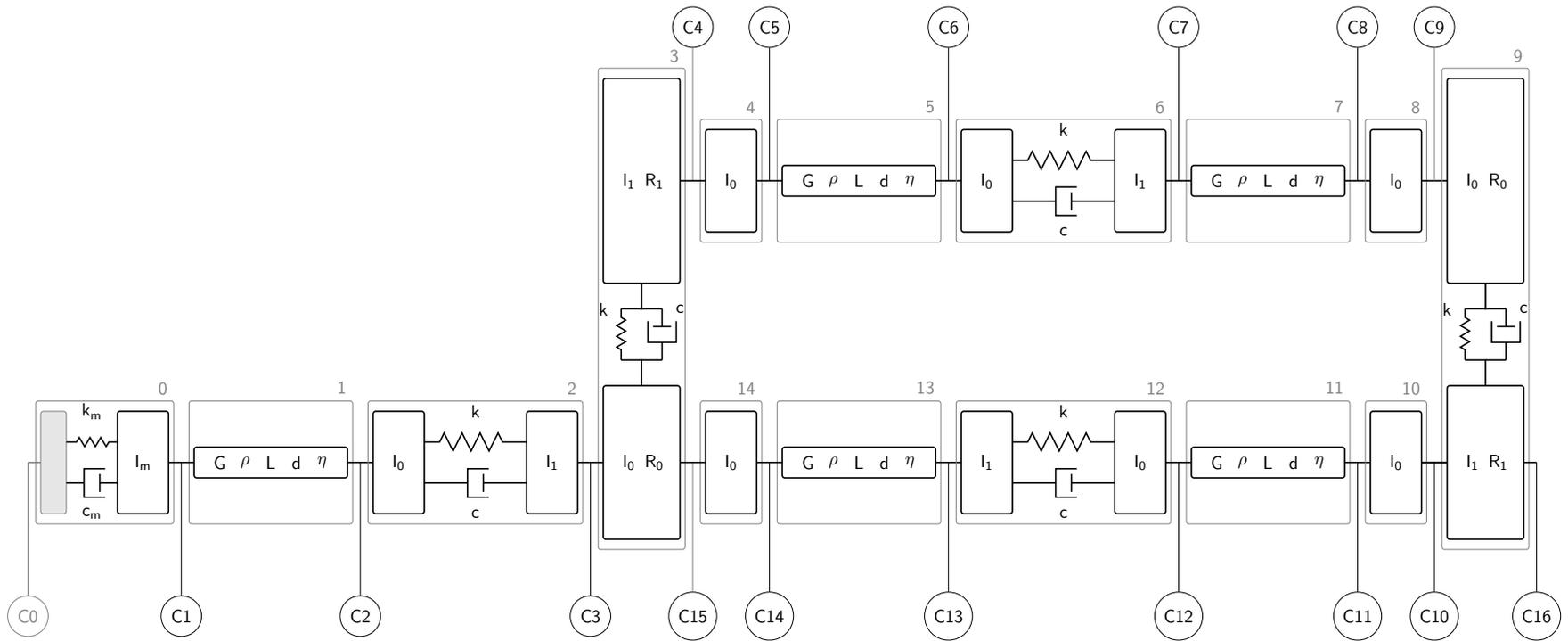
Figure 4.3: Frequency-based model of a passive single back-to-back torsional fatigue test rig. C# assigns a label at coordinates throughout the system. Components: servomotor 0; gearboxes 3 9; universal joint 4 8 10 14; continuous shaft 1 5 7 11 13; coupling 2 6 12.
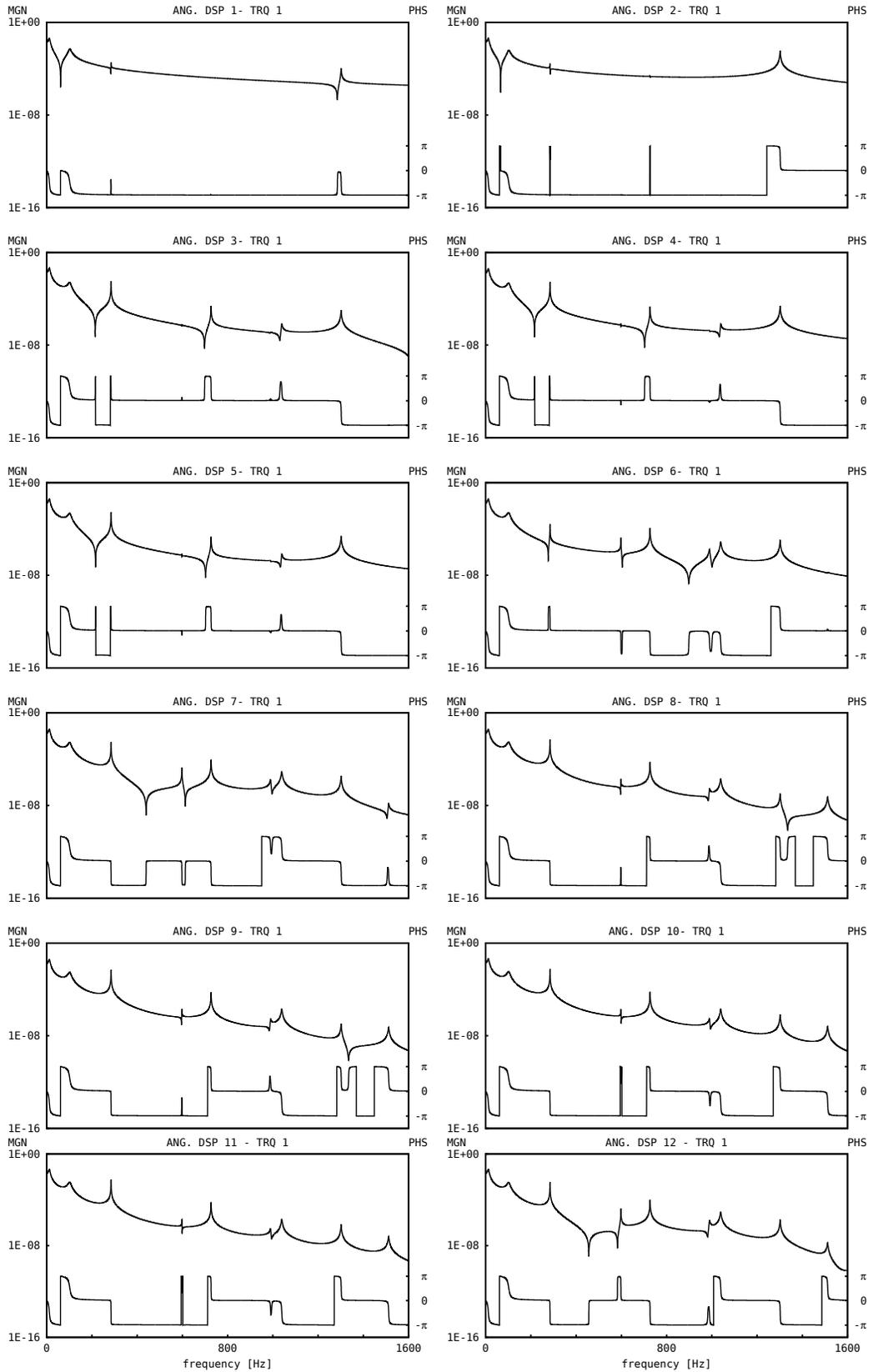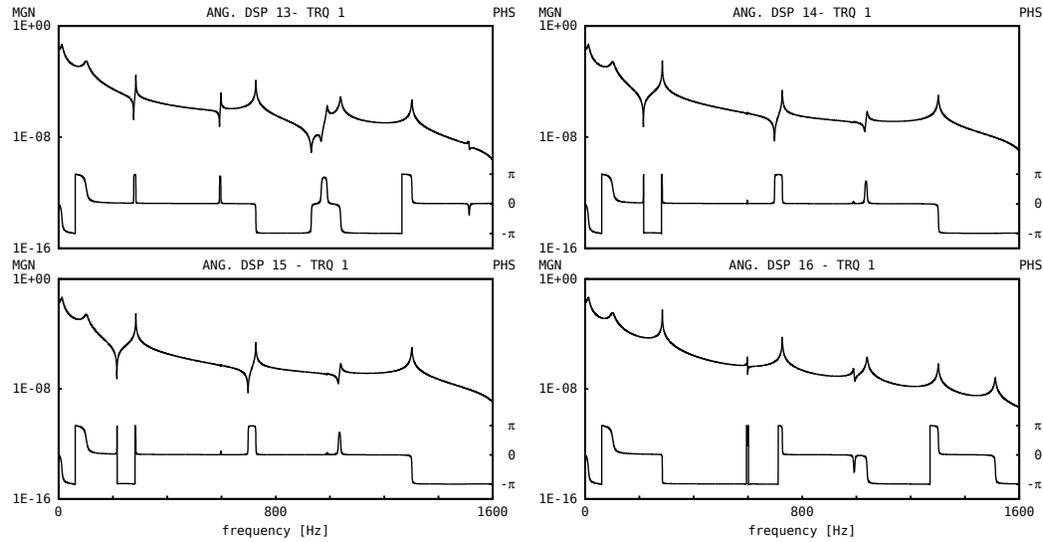
Figure 4.4: Predicted torsional frequency response functions for a torsional excitation at coordinate 1. N.B. ANG. DSP # refers to angular displacement at coordinate C# of Fig. 4.3; TRQ # to external torque applied at coordinate C#.

predictions of the dynamic behaviour of this arrangement have been obtained and illustrated in Fig. 4.5. functions. As expected, the torsional natural frequencies do not change appreciably. Applying the relationship between input-voltage and abutment displacement, the tuned model could also be used to simulate the torsional deflected shapes. It would be of interest to develop such a model in subsequent investigations; however, based on the current status, the approach of Leishman *et al.* is the most coherent for a complete investigation comprising the mode shapes in the frequency domain, Fig. 4.6.

Adding the extra inertia with which the universal joint is modelled to the relative gear inertia has observable differences for torsional natural frequencies, listed in Tab.4.2, and torsional deflected shapes; however, their values and trends can be considered comparable to the former ones.

Again, due to the similarity of both models, analogous observations regarding the parameter influences of sub-systems on rig behaviours have been obtained. The reader is directed to the work of Leishman *et al.* [5] for more in-depth information.

In order to determine the torsional vibration modes of the rotating rig, angular displacement at system coordinate 1 has been set to magnitude 1. Therefore, direct receptance α_{11} has permitted the derivation of the torque required to produce this displacement at the same coordinate. Finally, the response of each coordinate has been calculated using the second technique described in Section 2.5. Figure 4.6 shows the simulated torsional modes.
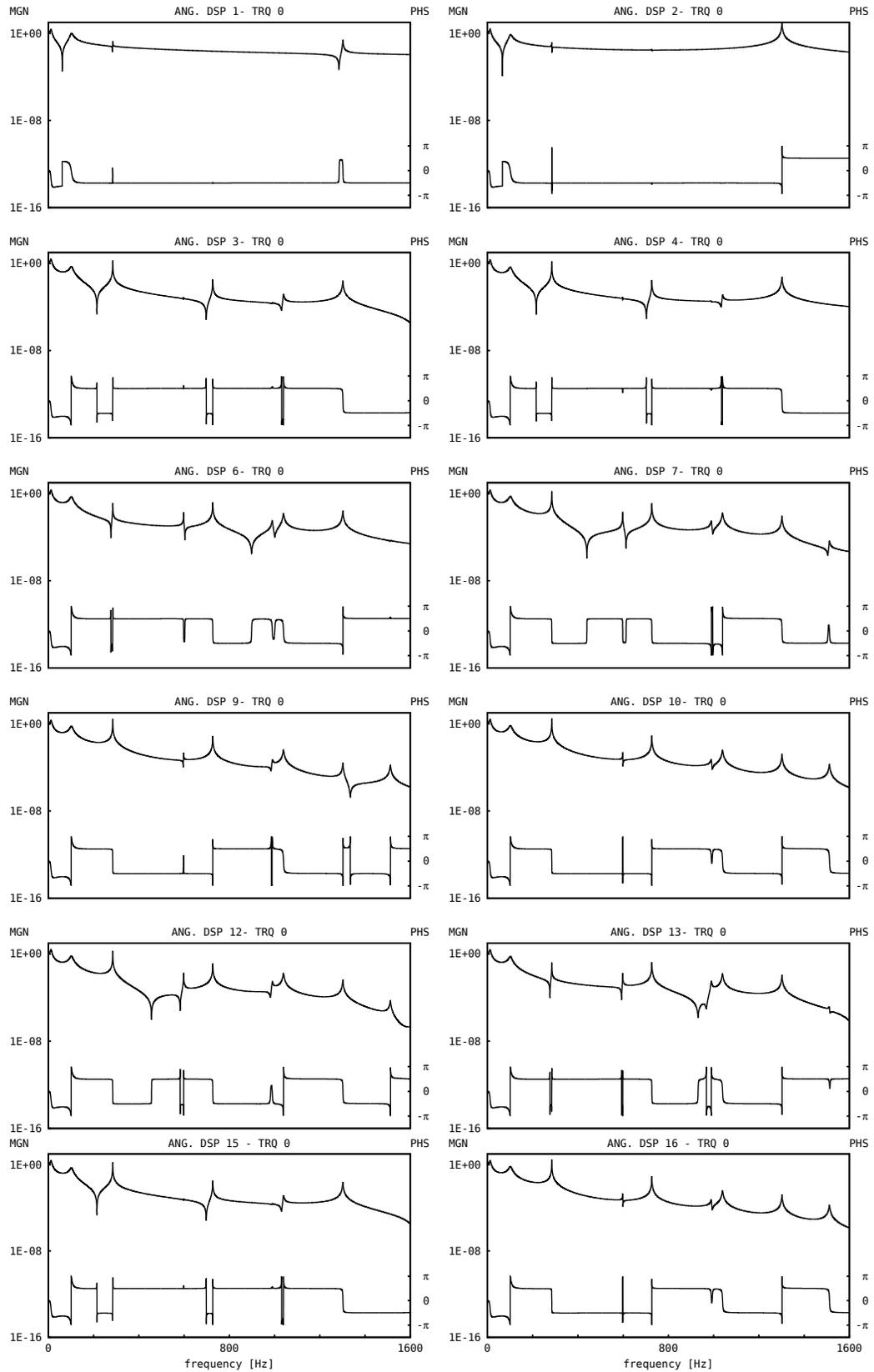
Figure 4.5: Abutment excitation: predicted torsional frequency response functions. N.B. ANG. DSP #
refers to angular displacement at coordinate C# of Fig. 4.3; TRQ # to external torque
applied at coordinate C#.

Table 4.2: Simulated torsional natural frequencies of simple back-to-back rig

						Frequency [Hz]		
12.	102.	285.	598.	727.	989.	1039.	1302.	1512.
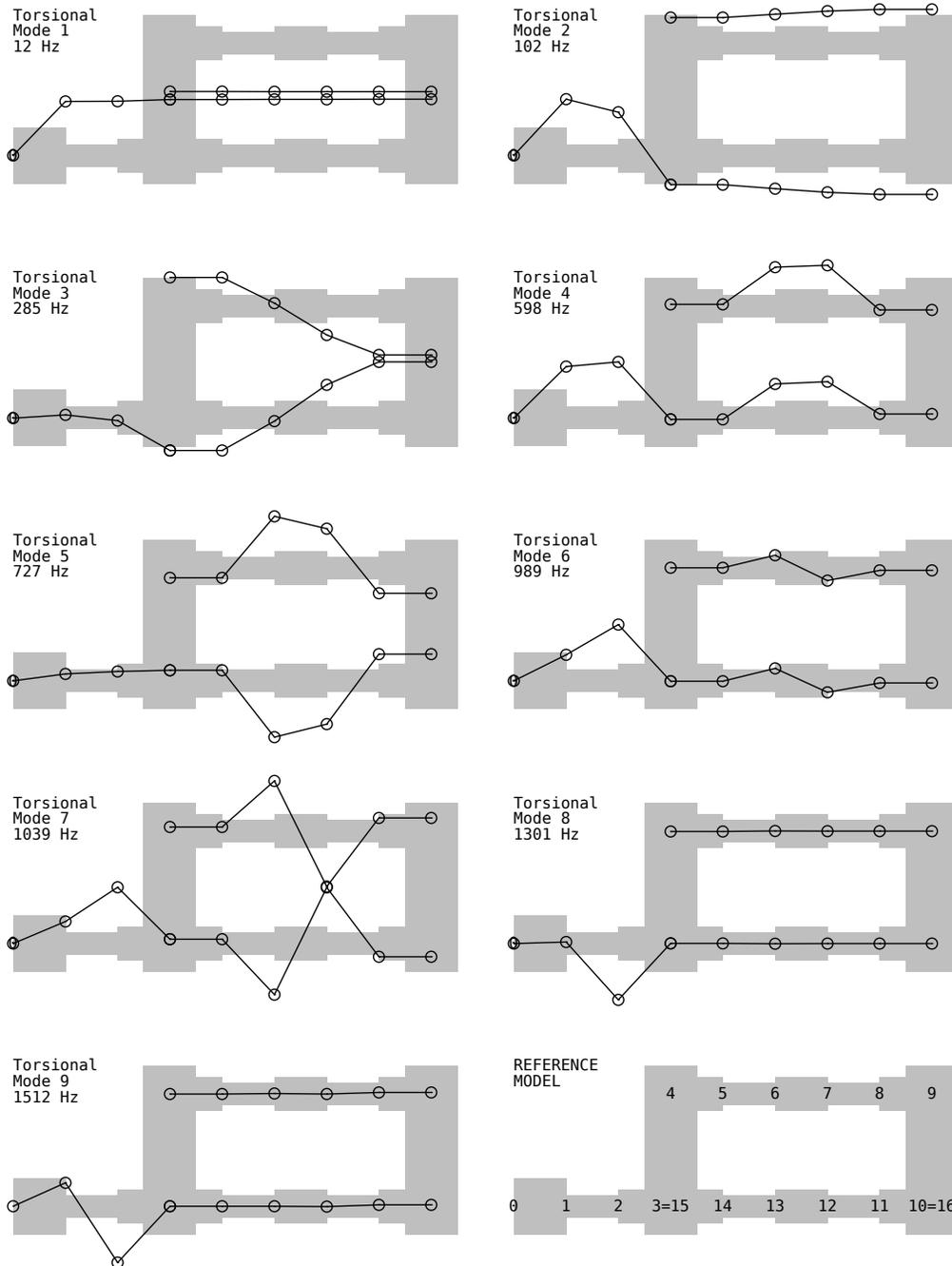
Figure 4.6: Single back-to-back rig torsional deflected shapes.

The 0 mode corresponds to pure rotational motion and has not been reported. To facilitate

Table 4.3: Single back-to-back torsional fatigue test rig: used parameters in the simulations. 'Sys' refers to system numberical label of Fig. 4.3; 'Tp' to sub-system type; data system units S.I.

Sys	Tp	Symbols	Values
0	0	k_m c_m I_m	50. 0.3 0.002783
1	4	G ρ L d η	8.E+10 7800.0 0.25 0.02 0.017
2	2	I_0 k c I_1	0.000079 900.0 0.00008 0.000079
-	67		1 3 10
3	3	I_0 k c I_1	0.001172 0.041275 8.E+07 10.8 0.001758 0.047625
4	5	I_0	4.211E-05
5	4	G ρ L d η	8.E+10 7800.0 0.25 0.02 0.001
6	2	I_0 k c I_1	0.000310272 4000.0 0.003 .000310272
7	4	G ρ L d η	8.E+10 7800.0 0.25 0.02 0.001
8	5	I_0	4.211E-05
9	3	I_0 k c I_1	0.000651 0.047625 1.25E+07 2.7 0.000486 0.041275
10	5	I_0	4.211E-05
11	4	G ρ L d η	8.E+10 7800.0 0.25 0.02 0.001
12	2	I_0 k c I_1	0.000310272 4000.0 0.003 0.000310272
13	4	G ρ L d η	8.E+10 7800.0 0.25 0.02 0.001
14	5	I_0	4.211E-05
-	67		0

visualisation of the deflected and mode shapes of the system, background representations of the rig in Fig. 4.6 do not show the misalignment angle information ($\delta = 30°$). As reported by Leishman *et al.* [5], the third mode (285 Hz) has been shown to be particularly dominant in all the predictions. Due to the specimen position between coordinates 6 and 7 (12 and 13), the 285 Hz mode, were it to be excited, would introduce a non-negligible stress cycle superimposed on the static one. Simulations have also shown that internal moments relative to the 7th mode at 1039 Hz can distort the predicted torsional cycles deriving from the static model; however, in working conditions excitation at 1039 Hz is unlikely to occur. A different behaviour would be observed at a frequency of 285 Hz if the secondary resonance phenomenon, associated with the universal joint non-linearities were considered (Section 3.3.1.4). In fact, it is indicated that, for a servomotor speed of 3000 rpm (50 Hz), superpositioning of system responses with those of the excitation frequency would feed the resonance frequency at 285 Hz. Due to lack of experimental data, both in the frequency and time domains, is it only a syllogism; however, more investigations should be conducted in this direction. Tests on a real system are of particular relevance when considering start-up and run-down frequencies associated with any real fatigue tests using such a rig. During these phases there is the obvious potential to excite system frequencies below the final mean operating speed of 3000rpm. Time-domain models may allude to such problems; however, detailed investigations of this behaviour have not been undertaken due to other drawbacks associated with this arrangement. Both the static and dynamic models of a passive single back-to-back rig have been developed and investigated. The system has presented several interesting

characteristics, amongst which is the construction simplicity; however, the dependence of the mean torque value on the amplitude of the stress cycle applied on the specimen, inserted along one of the inclined shafts, (b,1) or (b,2), results in an important limitation. Increasing the mean value of the torsional oscillation via an external pre-load, θ_{PL}, also increases the amplitude of the torsional stress cycle.

4.2   Double back-to-back rig

In order to overcome the aforementioned limitations, Guzzomi *et al.* [1] developed a concept design of a passive double back-to-back torsional fatigue test arrangement and formulated its theoretical static model [2]. Figure 4.2.1 depicts this rig.

As mentioned in Chapter 1, these types of rigs can perform torsional fatigue tests in short times. Consequently, because of the significant loading speeds and the rig complexity, a dynamic model is necessary to predict more realistic arrangement behaviour.

4.2.1   Dynamic analysis

Using the receptance technique, a frequency-based dynamic rig model, consisting of both the lumped-mass and continuous components, has been derived and is illustrated in Fig. 4.8. The difference between left and right gearbox specifications, employed in the previous model for verification of the numerical program, is no longer relevant, so identical specifications have been adopted for each gearbox. Table 4.4 lists all parameters used in the input form required by the developed program, App. C.1.

Both of the locked-in back-to-back torques, caused by two external pre-loads, θ_{PL}, through the couplings, do not influence the frequency-based modelling of the system. In fact, due to the assumptions of linear and time-invariant systems, these steady stresses simply shift the mean torque value of the torsional vibration. In the time-domain model, however, these factors must be considered. Exciting the system of Fig. 4.8 with a torque applied at coordinate (C1) and simulating its responses at each coordinate, torsional fatigue test rig receptances have been calculated. Magnitude and phase results are presented in Figure 4.9. Twelve torsional natural frequencies have been found, values of which are listed in Tab. 4.5.

Several simulations, performed in the frequency domain, have lead to an investigation into the effects of component stiffness variations on the system dynamic behaviour. With reference to Fig. 4.8, these variations have been applied to element 13, representative of the test specimen, elements 20 - 23, representative of the multi-component shaft, and elements 9 and 2.

Table 4.4: Double back-to-back torsional fatigue test rig: used parameters in the simulations. 'Sys' refers to system numberical label of Fig. 4.8; 'Tp' to sub-system type; data system units S.I.

Sys	Tp	Symbols	values
0	0	k_m c_m I_m	50.0 0.3 .002783
1	4	G ρ L d η	8.E+10 7800.0 0.25 0.02 0.017
2	2	I_0 k c I_1	.000079 900.0 0.00008 0.000079
-	67	1 3 10	
3	3	I_0 k c I_1	0.001172 0.041275 8.E+07 10.8 0.001758 0.047625
4	5	I_0	4.211E-05
5	4	G ρ L d η	8.E+10 7800.0 0.1 0.02 0.017
6	5	I_0	4.211E-05
7	3	I_0 k c I_1	0.001172 0.041275 8.E+07 10.8 0.001758 0.047625
8	4	G ρ L d η	8.E+10 7800.0 0.2 0.02 0.017
9	2	I_0 k c I_1	0.000310272 4000.0 0.003 0.000310272
10	4	G ρ L d η	8.E+10 7800.0 0.2 0.02 0.017
11	3	I_0 k c I_1	0.001758 0.047625 8.E+07 10.8 0.001172 0.041275
12	4	G ρ L d η	8.E+10 7800.0 0.2 0.02 0.017
13	2	I_0 k c I_1	0.000310 4000.0 0.003 0.000310
14	4	G ρ L d η	8.E+10 7800.0 0.2 0.02 0.017
-	67		0
15	5	I_0	4.211E-05
16	4	G ρ L d η	8.E+10 7800.0 0.1 0.02 0.017
17	5	I_0	4.211E-05
18	3	I_0 k c I_1	0.001758 0.047625 8.E+07 10.8 0.001172 0.041275
19	4	G ρ L d η	8.E+10 7800.0 0.5 0.03 0.017
20	2	I_0 k c I_1	0.000310 4000.0 0.003 0.000310
21	4	G ρ L d η	8.E+10 7800.0 0.5 0.03 0.017
-	67		0

Changing the stiffness of sub-system 13, k_{13}, from 4000 to 2000 Nm/rad, lead to no changes in the 1st, 4th, 6th, 7th, 8th, 11th and 12th natural frequencies and a slight change in the 2nd. Reducing the stiffness of the continuous shafts 20 and 22 ($d_{20,22}$ from 30 mm to 20 mm) lead to no changes in the 6th, 9th and 12th natural frequencies and negligible changes in the 1st, 2nd and 10th. Increasing the stiffness of sub-system 2, k_2, from 900 to 2000 Nm/rad, lead to small, significant and large changes in the 2nd, 3rd and 12th natural frequencies, respectively. Reducing the stiffness of k_9 lead to little impact on the resonance frequencies; however, a small changes in the 3rd, 5th and 9th resonance frequencies were observed.

Figure 4.7: For readers' convenience reproduction of test rig schematic: double back-to-back system [2].
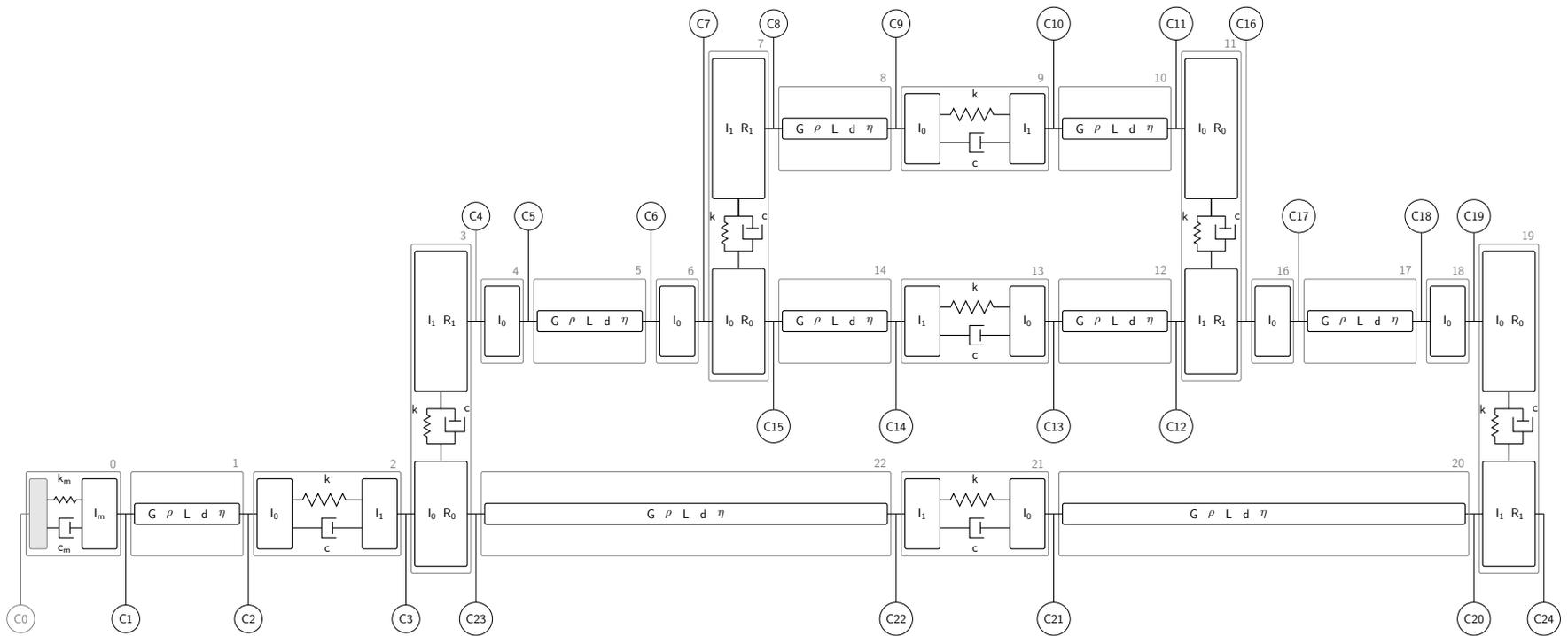
Figure 4.8: Frequency-based model of a passive double back-to-back torsional fatigue test rig. C# assigns a label at coordinates throughout the system. Components: servomotor 0; gearboxes 3 7 11 19; universal joint 4 6 16 18; continuous shaft 1 5 8 10 12 14 17 20 22; coupling 2 9 13 21
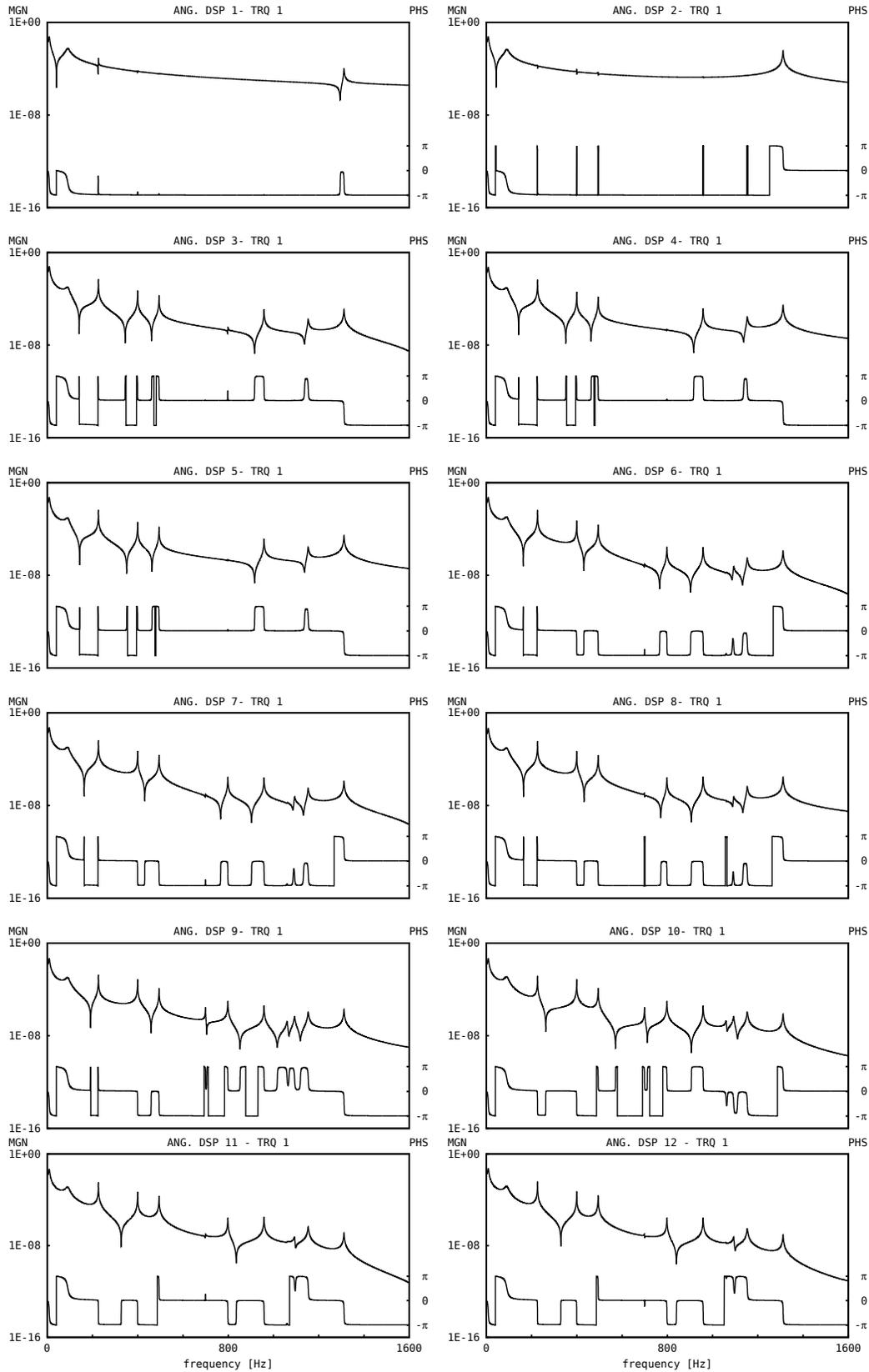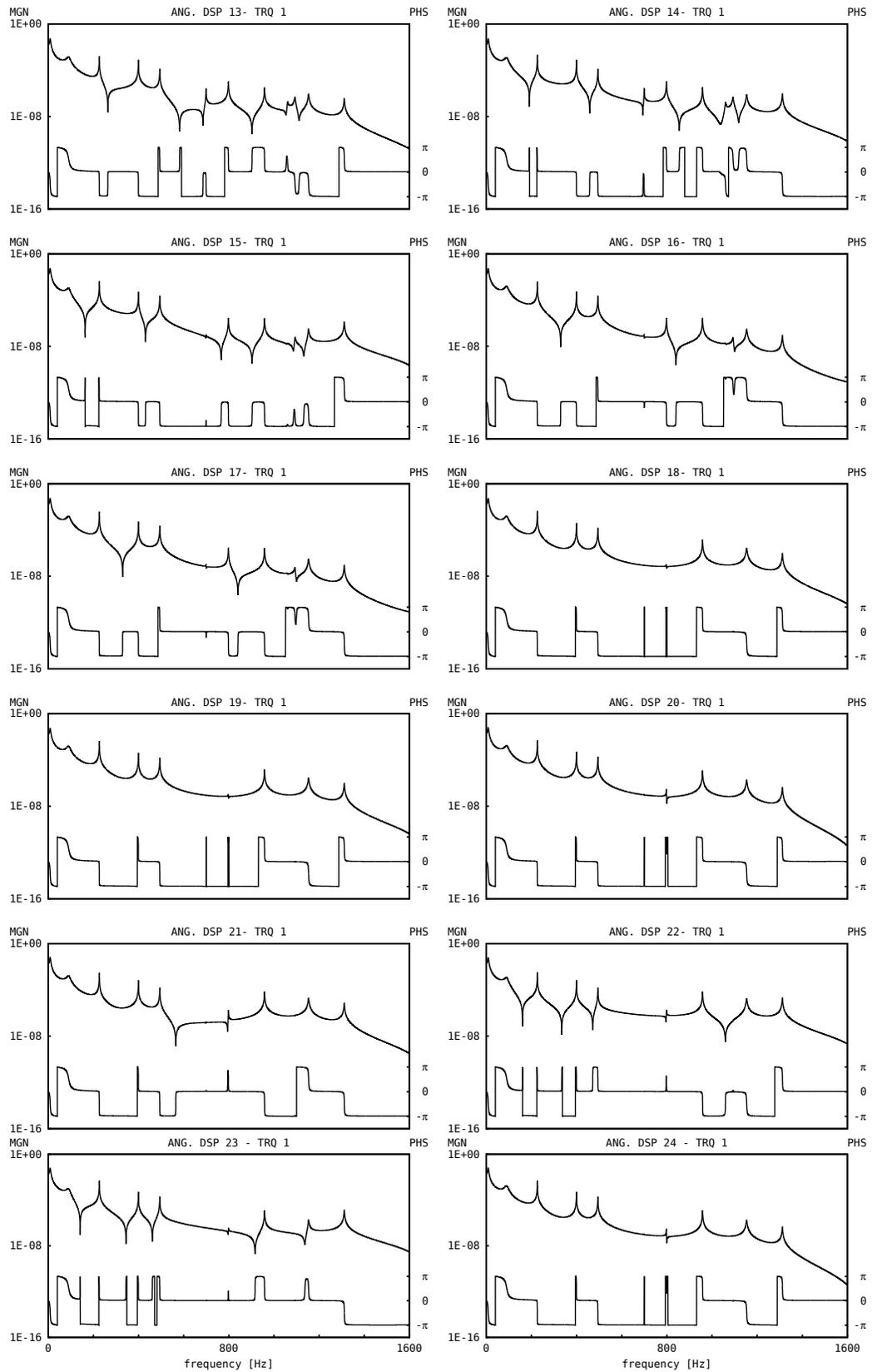
Figure 4.9: Simulated torsional frequency response functions for a double back-to-back system. N.B. ANG. DSP # refers to angular displacement at coordinate C# of Fig. 4.3; TRQ # to external torque applied at coordinate C#.

REFERENCE
MODEL

8   9   10   11

4   5   6   7=15   14   13   12=16   17   18   19

0   1   2   3=23   22   21   20=24

Torsional
Mode 1
9 Hz

Torsional
Mode 2
90 Hz

Torsional
Mode 3
226 Hz

Torsional
Mode 4
399 Hz

Torsional
Mode 5
494 Hz

Torsional
Mode 6
699 Hz

Torsional
Mode 7
798 Hz

Torsional
Mode 8
958 Hz

Torsional
Mode 9
1060 Hz

Torsional
Mode 10
1093 Hz

Torsional
Mode 11
1153 Hz
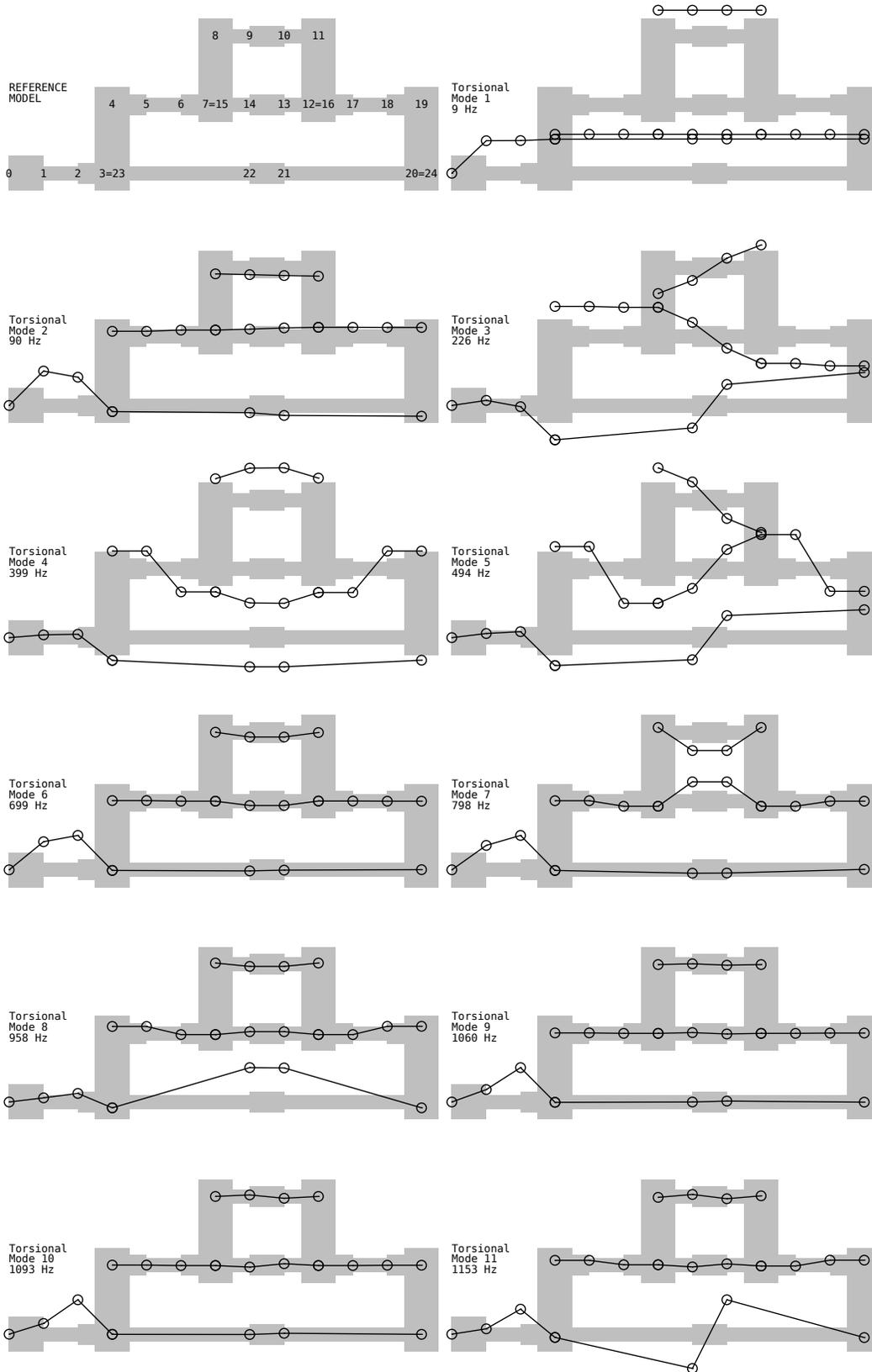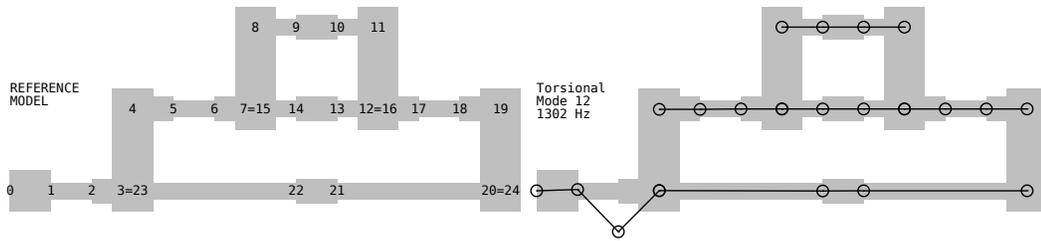
Figure 4.10: Double back-to-back rig torsional deflected shapes.

Figure 4.10 displays deflected shapes of the investigated arrangement. Again, background visualisations of the rig lack the misalignment angle data relative to the double universal joint driveline pairs (typically 30°). It is of interest to note that the concept design by Guzzomi *et al.* [2] also allows the misalignment angle, δ, to be modified so that a secondary amplitude control can be carried out on the torsional cyclic stresses acting on the specimen; however, due to the low influence of δ on the frequency model universal joint employed in rotating machines, Equation (3.57), negligible dynamic variations of rig for a broad range of δ from 0° to 45°, can be observed. On the other hand, a time domain analysis should display an increase of non-linear effects, due to growing values of the Fourier Series even order components for the universal joint (Chapter 3). The 399 Hz mode dominates the simulations, but should not interfere with the cyclic load acting on the specimen. On the other hand, important effects on the specimen could result from the 3rd and 5th modes (Figure 4.10). Again, considering a servomotor rotational speed of 3000 rpm, 2nd order servomotor sideband energy may feed the resonance frequency at 226 Hz, inducing the 3th mode. Though the extent to which this may be a problem would depend on the damping levels, the width of the resonance peak and the position and value of this resonant frequency it is obviously a function of the simulation parameters used. As stated previously, the single loop system start-up and run-down frequencies could so too be a problem for this new arrangement. Without doubt, in order to confirm these scenarios and quantify responses at coordinates throughout the system, a more in-depth analysis would need to be realised in the time domain and ultimately confirm by experiment.

Table 4.5: Simulated torsional natural frequencies of double back-to-back rig

											frequency [Hz]
9.	89.	226.	399.	494.	699.	798.	958.	1059.	1092	1153	1311

5 | Conclusions and Future Work

This dissertation focussed on innovative applications of the universal joint as a torsional excitation device and its implementation in passive back-to-back torsional fatigue test rigs of Guzzomi*et al.* [1, 2]. In particular the thesis looked at the dynamics of both the joint and the rigs in detail. A brief overview of the principal findings from this dissertation are presented below.

Chapter 2 presented some essential concepts of the receptance method. Some new approaches where presented, facilitating the modelling of 4-node elements. A numerical model was developed, allowing interconnection of systems with two coordinates and closed multi-loop schemes. The model calculates the receptance functions, modal and deflected shapes and internal strain acting on sub-system components of a complete, general system.

Chapter 3 developed an investigation into components required to model passive torsional fatigue test rigs. In particular, detailed analysis of a universal joint was undertaken, presenting both its dynamic model and investigations into its variable inertia. The joint was modelled first as a rigid-body two-inertia system, comprising a massless crosspiece and no friction forces. A rigid-body three-inertia system model with no friction forces was then developed. In the latter model the floating element and its gyroscopic effects were considered. Using two different kinematic assumptions, torsional receptances for each universal joint model were developed. Due to a number of interesting attributes evidenced during the model derivation, investigations into the behaviour of each were conducted both in the frequency and time domains.

Both models confirm that the inertia variation is a function of the misalignment angle and angular position. Changes in the inertia with angular position and misalignment angle were presented and discussed. The inertia variation associated with an inclined joint is not trivial. An in-depth investigation of its characteristics indicated that a second order cosine for misalignment angles typical of operating conditions ($\leq 30°$) approximates the inertia variation with a discrepancy within

1.6%. For increasing misalignment angles, an accurate description of the variation requires more even order cosine terms. Systems with inertia variation are known to exhibit non-linear frequency coupling between rotational speed and average natural frequencies. The same is true of systems with universal joints, which has been demonstrated by the novel application to engine balancing. Frequency spectra show significant reductions of non-linear behaviour for a single-cylinder reciprocating engine combined with a tuned flywheel (universal joint + Inertia), in contrast to a standard flywheel (inertia). Further to these developments, it would be interesting to analyse lateral vibration effects induced by this novel application.

Good agreement was obtained experimentally for the two-inertia model by using a developed torsional rig, which adopted a novel method of applying a locked-in torque to an open system. The more advanced three-inertia model for a universal joint, derived from a dual algebra formulation of a spherical four-link mechanism in closed form, showed some extra effects brought about by the dynamics of the inertial cross-piece. Although these effects are often small, and for most applications encourage use of the two-inertia model, the three-inertia model derivation permits the inclusion of contact damping and stiffness of its kinematic pairs, though these complexities were not incorporated in this study. With formulations presented in Chapter 2, a 3D receptance model for the joint could be derived.

Finally, Chapter 5 dealt with a detailed dynamic and mechanical design analysis of the passive back-to-back torsional fatigue test rigs, innovatively designed by Guzzomi*et al.* [1, 2].

For the single back-to-back arrangement, both static and dynamic models were developed. Using the static model, two rig layouts, depending on the phase between the pairs of double universal joint drivelines, were discussed. It was shown that phasing of $90°$ reduces the torsional loading cycle amplitude acting on the gearboxes, reducing to zero for gearbox ratios set to 1. This would thus be the preferred arrangement.

Using the receptance method, a frequency-based dynamic model of the single back-to-back rig was derived. It consists of lumped-mass sub-systems and continuous shafts. Universal joints were modelled as inertias set to the average value of the two-inertia model for a misalignment angle of $30°$. Simulated frequency response functions, both magnitude and phase, at coordinates throughout the system for a servomotor excitation (torque excitation, fixed abutment) over a range from 0 to 1600 Hz match results available in the literature for a system with similar inertial, stiffness and damping properties, validating the developed source code. Using a servomotor receptance model (abutment excitation),the new predicted receptances better correspond to measured FRFs reported in the literature. It would thus be of interest to investigate the relationship between

input-voltage excitation and abutment displacement in order to simulate deflected and mode shapes with a more physical excitation model. Deflected and mode shapes for the torque excitation case were also predicted. For the component parameters used in the simulations, the third mode, at frequency of 285 Hz, is particularly dominant. This mode is of concern for its potential to induce significant vibrational stress under proposed operational speeds. For a servomotor speed of 3000 rpm, superpositioning of the system responses with those of excitation frequency could feed the resonance frequency at 285 Hz. Furthermore, considering servomotor start-up and run-down phases, the 285 Hz mode may be excited. More in-depth analysis in the time domain and experiments are recommended directions for future research. Similar investigations in the frequency domain were also undertaken for the second passive back-to-back torsional fatigue test rig. Twelve torsional resonance frequencies were found over a range from 0 to 1600 Hz. The influences of system parameter on FRFs were discussed and predicted mode shapes were derived. Mode shapes depict well the complex dynamic behaviour of the system. The dominant mode is the 4th at 399 Hz; however, it likely would not interfere with cycling load acting on the specimen. Two mode shapes of concern were found: the 3rd and the $5^{t}h$ modes at 224Hz and 492 Hz, respectively, as these could be potentially excited during operation to and from the operating speed of 3000rpm.

Appendices

A |

A.1 Matrices $[L]$, $[\Lambda]$ and $[N]$

Dual components of $[L]$:

Note: dual part of $\hat{l}_{\xi\zeta}$ is zero

$$(l_{11})_r = C\alpha_{12}C\theta_2 \qquad (l_{12})_r = -C\alpha_{12}S\theta_2 \qquad (l_{13})_r = S\alpha_{12} \qquad \text{(A.1a)}$$

$$(l_{21})_r = S\theta_2 \qquad (l_{22})_r = C\theta_2 \qquad (l_{23})_r = 0 \qquad \text{(A.1b)}$$

$$(l_{31})_r = -S\alpha_{12}C\theta_2 \qquad (l_{32})_r = S\alpha_{12}S\theta_2 \qquad (l_{33})_r = C\alpha_{12} \qquad \text{(A.1c)}$$

Dual components of $[\Lambda]$:

Note: dual part of $\hat{\lambda}_{\xi\zeta}$ is zero

$$(\lambda_{11})_r = C\alpha_{12}(-C\alpha_{34}S\theta_2 S\theta_3 + C\alpha_{23}C\alpha_{34}C\theta_2 C\theta_3 - S\alpha_{23}S\alpha_{34}C\theta_2)$$
$$+ S\alpha_{12}(-S\alpha_{23}C\alpha_{34}C\theta_3 - C\alpha_{23}S\alpha_{34}) \qquad \text{(A.2a)}$$

$$(\lambda_{12})_r = C\alpha_{12}(-C\alpha_{23}C\theta_2 S\theta_3 - S\theta_2 C\theta_3) + S\alpha_{12}S\alpha_{23}S\theta_3 \qquad \text{(A.2b)}$$

$$(\lambda_{13})_r = C\alpha_{12}(-S\alpha_{34}S\theta_2 S\theta_3 + C\alpha_{23}S\alpha_{34}C\theta_2 C\theta_3 + S\alpha_{23}C\alpha_{34}C\theta_2)$$
$$+ S\alpha_{12}(C\alpha_{23}C\alpha_{34} - S\alpha_{23}S\alpha_{34}C\theta_3) \qquad \text{(A.2c)}$$

$$(\lambda_{21})_r = C\alpha_{34}C\theta_2 S\theta_3 + C\alpha_{23}C\alpha_{34}S\theta_2 C\theta_3 - S\alpha_{23}S\alpha_{34}S\theta_2 \qquad \text{(A.3a)}$$

$$(\lambda_{22})_r = C\theta_2 C\theta_3 - C\alpha_{23}S\theta_2 S\theta_3 \qquad \text{(A.3b)}$$

$$(\lambda_{23})_r = S\alpha_{34}C\theta_2 S\theta_3 + C\alpha_{23}S\alpha_{34}S\theta_2 C\theta_3 + S\alpha_{23}C\alpha_{34}S\theta_2 \qquad \text{(A.3c)}$$

$$(\lambda_{31})_r = -C\alpha_{12}(S\alpha_{23}C\alpha_{34}C\theta_3 + C\alpha_{23}S\alpha_{34})$$

$$- S\alpha_{12}(-C\alpha_{34}S\theta_2 S\theta_3 + C\alpha_{23}C\alpha_{34}C\theta_2 C\theta_3 - S\alpha_{23}S\alpha_{34}C\theta_2) \tag{A.4a}$$

$$(\lambda_{32})_r = C\alpha_{12}S\alpha_{23}S\theta_3 + S\alpha_{12}(C\alpha_{23}C\theta_2 S\theta_3 + S\theta_2 C\theta_3) \tag{A.4b}$$

$$(\lambda_{33})_r = C\alpha_{12}(C\alpha_{23}C\alpha_{34} - S\alpha_{23}S\alpha_{34}C\theta_3)$$

$$- S\alpha_{12}(C\alpha_{23}S\alpha_{34}C\theta_2 C\theta_3 + S\alpha_{23}C\alpha_{34}C\theta_2 - S\alpha_{34}S\theta_2 S\theta_3) \tag{A.4c}$$

Dual components of $[N]$:

$$\hat{n}_{11} = (n_{11})_r + \epsilon R(n_{11})_d \tag{A.5a}$$

$$(n_{11})_r = C\alpha_{12}(-C\alpha_{34}S\theta_2 S\theta_3 + C\alpha_{23}C\alpha_{34}C\theta_2 C\theta_3 - S\alpha_{23}S\alpha_{34}C\theta_2)$$

$$+ S\alpha_{12}(-S\alpha_{23}C\alpha_{34}C\theta_3 - C\alpha_{23}S\alpha_{34})$$

$$(n_{11})_d = -C\alpha_{12}(S\theta_2 C\theta_3 + C\alpha_{23}C\theta_2 S\theta_3) + S\alpha_{12}S\alpha_{23}S\theta_3$$

$$\hat{n}_{12} = (n_{12})_r + \epsilon R(n_{12})_d \tag{A.5b}$$

$$(n_{12})_r = C\alpha_{12}(-C\alpha_{23}C\theta_2 S\theta_3 - S\theta_2 C\theta_3) + S\alpha_{12}S\alpha_{23}S\theta_3$$

$$(n_{12})_d = C\alpha_{12}(S\alpha_{23}S\alpha_{34}C\theta_2 - C\alpha_{23}C\alpha_{34}C\theta_2 C\theta_3 + C\alpha_{34}S\theta_2 S\theta_3)$$

$$+ S\alpha_{12}(C\alpha_{23}S\alpha_{34} + S\alpha_{23}C\alpha_{34}C\theta_3)$$

$$\hat{n}_{13} = (n_{13})_r + \epsilon R(n_{13})_d \tag{A.5c}$$

$$(n_{13})_r = C\alpha_{12}(-S\alpha_{34}S\theta_2 S\theta_3 + C\alpha_{23}S\alpha_{34}C\theta_2 C\theta_3 + S\alpha_{23}C\alpha_{34}C\theta_2)$$

$$+ S\alpha_{12}(C\alpha_{23}C\alpha_{34} - S\alpha_{23}S\alpha_{34}C\theta_3)$$

$$(n_{13})_d = 0$$

$$\hat{n}_{21} = (n_{21})_r + \epsilon R(n_{21})_d \tag{A.6a}$$

$$(n_{21})_r = C\alpha_{34}C\theta_2 S\theta_3 + C\alpha_{23}C\alpha_{34}S\theta_2 C\theta_3 - S\alpha_{23}S\alpha_{34}S\theta_2$$

$$(n_{21})_d = C\theta_2 C\theta_3 - C\alpha_{23} S\theta_2 S\theta_3$$

$$\hat{n}_{22} = (n_{22})_r + \epsilon R(n_{22})_d \tag{A.6b}$$

$$(n_{22})_r = C\theta_2 C\theta_3 - C\alpha_{23} S\theta_2 S\theta_3$$

$$(n_{22})_d = S\alpha_{23} S\alpha_{34} S\theta_2 - C\alpha_{23} C\alpha_{34} S\theta_2 C\theta_3 - C\alpha_{34} C\theta_2 S\theta_3$$

$$\hat{n}_{23} = (n_{23})_r + \epsilon R(n_{23})_d \tag{A.6c}$$

$$(n_{23})_r = S\alpha_{34} C\theta_2 S\theta_3 + C\alpha_{23} S\alpha_{34} S\theta_2 C\theta_3 + S\alpha_{23} C\alpha_{34} S\theta_2$$

$$(n_{23})_d = 0$$

$$\hat{n}_{31} = (n_{31})_r + \epsilon R(n_{31})_d \tag{A.7a}$$

$$(n_{31})_r = -C\alpha_{12}(S\alpha_{23} C\alpha_{34} C\theta_3 + C\alpha_{23} S\alpha_{34})$$
$$+ S\alpha_{12}(C\alpha_{34} S\theta_2 S\theta_3 - C\alpha_{23} C\alpha_{34} C\theta_2 C\theta_3 + S\alpha_{23} S\alpha_{34} C\theta_2)$$

$$(n_{31})_d = C\alpha_{12} S\alpha_{23} S\theta_3 + S\alpha_{12}(S\theta_2 C\theta_3 + C\alpha_{23} C\theta_2 S\theta_3)$$

$$\hat{n}_{32} = (n_{32})_r + \epsilon R(n_{32})_d \tag{A.7b}$$

$$(n_{32})_r = C\alpha_{12} S\alpha_{23} S\theta_3 + S\alpha_{12}(C\alpha_{23} C\theta_2 S\theta_3 + S\theta_2 C\theta_3)$$

$$(n_{32})_d = C\alpha_{12}(C\alpha_{23} S\alpha_{34} + S\alpha_{23} C\alpha_{34} C\theta_3)$$
$$+ S\alpha_{12}(C\alpha_{23} C\alpha_{34} C\theta_2 C\theta_3 - S\alpha_{23} S\alpha_{34} C\theta_2 - C\alpha_{34} S\theta_2 S\theta_3)$$

$$\hat{n}_{33} = (n_{33})_r + \epsilon R(n_{33})_d \tag{A.7c}$$

$$(n_{33})_r = C\alpha_{12}(C\alpha_{23} C\alpha_{34} - S\alpha_{23} S\alpha_{34} C\theta_3)$$
$$+ S\alpha_{12}(S\alpha_{34} S\theta_2 S\theta_3 - C\alpha_{23} S\alpha_{34} C\theta_2 C\theta_3 - S\alpha_{23} C\alpha_{34} C\theta_2)$$

$$(n_{33})_d = 0$$

A.2   Derivation of equation of motion

Note: S, C and Ct are used to denote $\sin$, $\cos$ and $\cot$ functions respectively.

$$B = \frac{S\alpha_{12}}{S\alpha_{23}S\theta_3}(S\theta_2 C\theta_3 + C\alpha_{23}C\theta_2 S\theta_3) \tag{A.8a}$$

$$= S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 \tag{A.8b}$$

$$C\alpha_{12} + B = C\alpha_{12} + \frac{S\alpha_{12}}{S\alpha_{23}S\theta_3}(S\theta_2 C\theta_3 + C\alpha_{23}C\theta_2 S\theta_3) \tag{A.9a}$$

$$= S\alpha_{12}(Ct\alpha_{12} + \frac{S\theta_2 Ct\theta_3}{S\alpha_{23}} + Ct\alpha_{23}C\theta_2) \tag{A.9b}$$

$$= -\frac{\dot{\theta}_2}{\dot{\theta}_1} \tag{A.9c}$$

Component J_1^{II}

$$-S\alpha_{12}J_1^{II}\left[-S\alpha_{12}(\ddot{\theta}_1 C^2\theta_2 - \dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2)\right]$$

$$+ S\alpha_{12}\left[-J_1^{II}(\dot{\theta}_1 C\alpha_{12} + \dot{\theta}_2)\dot{\theta}_1 S\alpha_{12}S\theta_2 C\theta_2\right]$$

$$- J_1^{II}\dot{\theta}_1^2 S^2\alpha_{12}S\theta_2 C\theta_2 B \tag{A.10a}$$

$$= J_1^{II}S^2\alpha_{12}(\ddot{\theta}_1 C^2\theta_2 - 2\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2 - \dot{\theta}_1^2 C\alpha_{12}S\theta_2 C\theta_2 - \dot{\theta}_1^2 S\theta_2 C\theta_2 B) \tag{A.10b}$$

$$= J_1^{II}S^2\alpha_{12}\left[\ddot{\theta}_1 C^2\theta_2 - 2\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2 - \dot{\theta}_1^2 S\theta_2 C\theta_2 (C\alpha_{12} + B)\right] \tag{A.10c}$$

$$= J_1^{II}S^2\alpha_{12}(\ddot{\theta}_1 C^2\theta_2 - 2\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2 + \dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2) \tag{A.10d}$$

$$= J_1^{II}S^2\alpha_{12}(\ddot{\theta}_1 C^2\theta_2 - \dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2) \tag{A.10e}$$

$$= J_1^{II}\ddot{\theta}_1 S^2\alpha_{12}C^2\theta_2 - J_1^{II}\dot{\theta}_1\dot{\theta}_2 S^2\alpha_{12}S\theta_2 C\theta_2) \tag{A.10f}$$

$$= J_1^{II}\ddot{\theta}_1 S^2\alpha_{12}C^2\theta_2 + \frac{1}{2}\left(-2J_1^{II}\dot{\theta}_1 \frac{d\theta_2}{dt}\frac{d\theta_1}{d\theta_1} S^2\alpha_{12}S\theta_2 C\theta_2\right) \tag{A.10g}$$

$$= J_1^{II}S^2\alpha_{12}C^2\theta_2\ddot{\theta}_1 + \frac{1}{2}\left(-2J_1^{II}S^2\alpha_{12}S\theta_2 C\theta_2 \frac{d\theta_2}{d\theta_1}\right)\dot{\theta}_1^2 \tag{A.10h}$$

$$= J_1^{II}S^2\alpha_{12}C^2\theta_2\ddot{\theta}_1 + \frac{1}{2}\left[\frac{d}{d\theta_1}\left(J_1^{II}S^2\alpha_{12}C^2\theta_2\right)\right]\dot{\theta}_1^2 \tag{A.10i}$$

Component J_2^{II}

$$-S\alpha_{12}\big[-J_2^{II}\dot{\theta}_1 S\alpha_{12} S\theta_2 C\theta_2(\dot{\theta}_1 C\alpha_{12}+\dot{\theta}_2)\big]$$
$$+S\alpha_{12}\big[J_2^{II}S\alpha_{12}(\ddot{\theta}_1 S^2\theta_2+\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2)\big]$$
$$-(-J_2^{II}\dot{\theta}_1^2 S^2\alpha_{12} S\theta_2 C\theta_2 B) \tag{A.11a}$$

$$=J_2^{II}S^2\alpha_{12}(\ddot{\theta}_1 S^2\theta_2+2\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2$$
$$+\dot{\theta}_1^2 C\alpha_{12} S\theta_2 C\theta_2+\dot{\theta}_1^2 S\theta_2 C\theta_2 B) \tag{A.11b}$$

$$=J_2^{II}S^2\alpha_{12}\big[\ddot{\theta}_1 S^2\theta_2+2\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2+\dot{\theta}_1^2 S\theta_2 C\theta_2\big(C\alpha_{12}+B\big)\big] \tag{A.11c}$$

$$=J_2^{II}S^2\alpha_{12}(\ddot{\theta}_1 S^2\theta_2+2\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2-\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2) \tag{A.11d}$$

$$=J_2^{II}S^2\alpha_{12}(\ddot{\theta}_1 S^2\theta_2+\dot{\theta}_1\dot{\theta}_2 S\theta_2 C\theta_2) \tag{A.11e}$$

$$=J_2^{II}\ddot{\theta}_1 S^2\alpha_{12} S^2\theta_2+J_2^{II}\dot{\theta}_1\dot{\theta}_2 S^2\alpha_{12} S\theta_2 C\theta_2) \tag{A.11f}$$

$$=J_2^{II}\ddot{\theta}_1 S^2\alpha_{12} S^2\theta_2+\frac{1}{2}\big(2J_2^{II}\dot{\theta}_1\frac{d\theta_2}{dt}\frac{d\theta_1}{d\theta_1}S^2\alpha_{12} S\theta_2 C\theta_2\big) \tag{A.11g}$$

$$=J_2^{II}S^2\alpha_{12} S^2\theta_2\ddot{\theta}_1+\frac{1}{2}\big(2J_2^{II}S^2\alpha_{12} S\theta_2 C\theta_2\frac{d\theta_2}{d\theta_1}\big)\dot{\theta}_1^2 \tag{A.11h}$$

$$=J_2^{II}S^2\alpha_{12} S^2\theta_2\ddot{\theta}_1+\frac{1}{2}\Big[\frac{d}{d\theta_1}\Big(J_2^{II}S^2\alpha_{12} C^2\theta_2\Big)\Big]\dot{\theta}_1^2 \tag{A.11i}$$

Component J_3^{II}

$$-S\alpha_{12}\big[J_3^{II}\dot{\theta}_1 S\alpha_{12} S\theta_2 C\theta_2(\dot{\theta}_1 C\alpha_{12}+\dot{\theta}_2)\big]$$
$$+S\alpha_{12}\big[J_3^{II}(\dot{\theta}_1 C\alpha_{12}+\dot{\theta}_2)\dot{\theta}_1 S\alpha_{12} S\theta_2 C\theta_2\big]$$
$$-\big[J_3^{II}(\ddot{\theta}_1 C\alpha_{12}+\ddot{\theta}_2)B\big] \tag{A.12a}$$

$$=-J_3^{II}(\ddot{\theta}_1 C\alpha_{12}+\ddot{\theta}_2)B \tag{A.12b}$$

$$=-J_3^{II}\bigg\{\ddot{\theta}_1 C\alpha_{12}-S\alpha_{12}\big[\ddot{\theta}_1\big(Ct\alpha_{12}+Ct\alpha_{23}C\theta_2\big)$$
$$+\dot{\theta}_1^2 S\theta_2\big(C\alpha_{12} Ct\alpha_{23}-S\alpha_{12} C\theta_2\big)\big]$$
$$-\frac{S\alpha_{34}}{S\alpha_{23}}\big[(\dot{\tau}\dot{\theta}_1+\tau\ddot{\theta}_1)C\theta_3-\tau^2\dot{\theta}_1^2 C\alpha_{23} S\theta_3\big]\bigg\}B \tag{A.12c}$$

$$=J_3^{II}\bigg\{\Big[-C\alpha_{12}+S\alpha_{12}\big(Ct\alpha_{12}+Ct\alpha_{23}C\theta_2\big)+\frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\Big]B\ddot{\theta}_1$$
$$+\Big[S\alpha_{12} S\theta_2\big(C\alpha_{12} Ct\alpha_{23}-S\alpha_{12} C\theta_2\big)$$

$$+ \frac{S\alpha_{34}}{S\alpha_{23}}\left(\frac{d\tau}{d\theta_1}C\theta_3 - \tau^2 C\alpha_{23}S\theta_3\right)\bigg]B\dot{\theta}_1^2\bigg\} \tag{A.12d}$$

$$= J_3^{II}\bigg\{\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)\frac{S\alpha_{12}}{S\alpha_{23}S\theta_3}(S\theta_2 C\theta_3 + C\alpha_{23}C\theta_2 S\theta_3)\ddot{\theta}_1$$

$$+ \left[S\alpha_{12}S\theta_2(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2) + \frac{S\alpha_{34}}{S\alpha_{23}}\left(\frac{d\tau}{d\theta_1}C\theta_3 - \tau^2 C\alpha_{23}S\theta_3\right)\right]$$

$$\cdot \frac{S\alpha_{12}}{S\alpha_{23}S\theta_3}(S\theta_2 C\theta_3 + C\alpha_{23}C\theta_2 S\theta_3)\dot{\theta}_1^2\bigg\} \tag{A.12e}$$

$$= J_3^{II}\bigg\{\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)^2\ddot{\theta}_1$$

$$+ (S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3)$$

$$\cdot \big[S\alpha_{12}S\theta_2(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2)$$

$$+ \frac{S\alpha_{34}}{S\alpha_{23}}(\frac{d\tau}{d\theta_1}C\theta_3 - \tau^2 C\alpha_{23}S\theta_3)\big]\dot{\theta}_1^2\bigg\} \tag{A.12f}$$

$$= J_3^{II}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)^2\ddot{\theta}_1$$

$$+ \frac{1}{2}\bigg\{2J_3^{II}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)$$

$$\cdot \left[S\alpha_{12}S\theta_2(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2) + \frac{S\alpha_{34}}{S\alpha_{23}}\left(\frac{d\tau}{d\theta_1}C\theta_3 - \tau^2 C\alpha_{23}S\theta_3\right)\right.$$

$$+ \tau\frac{S\alpha_{34}}{S\alpha_{23}}C\alpha_{23}S\theta_3\frac{d\theta_2}{d\theta_1} - \tau\frac{S\alpha_{34}}{S\alpha_{23}}C\alpha_{23}S\theta_3\frac{d\theta_2}{d\theta_1}\bigg]\bigg\}\dot{\theta}_1^2 \tag{A.12g}$$

$$= J_3^{II}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)^2\ddot{\theta}_1$$

$$+ \frac{1}{2}\bigg\{2J_3^{II}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)$$

$$\cdot \left[\frac{S\alpha_{34}}{S\alpha_{23}}\frac{d\tau}{d\theta_1}C\theta_3 - \frac{S\alpha_{34}}{S\alpha_{23}}\tau^2 C\alpha_{23}S\theta_3 + S\alpha_{12}S\theta_2(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2)\right.$$

$$+ \tau\frac{S\alpha_{34}}{S\alpha_{23}}C\alpha_{23}S\theta_3\frac{d\theta_2}{d\theta_1} - S\alpha_{12}Ct\alpha_{23}S\theta_2\frac{d\theta_2}{d\theta_1}\bigg]\bigg\}\dot{\theta}_1^2 \tag{A.12h}$$

$$= J_3^{II}\big(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\big)^2\ddot{\theta}_1$$

$$+ \frac{1}{2}\bigg\{2J_3^{II}\big(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\big)$$

$$\cdot \left[-S\alpha_{12}Ct\alpha_{23}S\theta_2\frac{d\theta_2}{d\theta_1} + \frac{S\alpha_{34}}{S\alpha_{23}}\frac{d\tau}{d\theta_1}C\theta_3 - \frac{S\alpha_{34}}{S\alpha_{23}}\tau^2 C\alpha_{23}S\theta_3\right.$$

$$+ \tau\frac{S\alpha_{34}}{S\alpha_{23}}C\alpha_{23}S\theta_3\frac{d\theta_2}{d\theta_1} + S\alpha_{12}S\theta_2(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2)\bigg]\bigg\}\dot{\theta}_1^2 \tag{A.12i}$$

$$= J_3^{II}\big(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\big)^2\ddot{\theta}_1$$

$$
+ \frac{1}{2} \Bigg\{ 2J_3^{II} \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}} \tau C\theta_3 \Big)
$$

$$
\cdot \Bigg[-S\alpha_{12} Ct\alpha_{23} S\theta_2 \frac{d\theta_2}{d\theta_1} + \frac{S\alpha_{34}}{S\alpha_{23}} \frac{d\tau}{d\theta_1} C\theta_3 - \frac{S\alpha_{34}}{S\alpha_{23}} \tau^2 C\alpha_{23} S\theta_3
$$

$$
+ \tau \frac{S\alpha_{34}}{S\alpha_{23}} C\alpha_{23} S\theta_3 \frac{d\theta_2}{d\theta_1} + S\alpha_{12} S\theta_2 C\alpha_{12} Ct\alpha_{23} \frac{S\alpha_{34} S\theta_3}{S\alpha_{34} S\theta_3}
$$

$$
- S\alpha_{12} S\theta_2 S\alpha_{12} C\theta_2 \frac{S\alpha_{34} S\theta_3 S\alpha_{23}}{S\alpha_{34} S\theta_3 S\alpha_{23}} \Bigg] \Bigg\} \dot{\theta}_1^2 \tag{A.12j}
$$

$$
= J_3^{II} \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}} \tau C\theta_3 \Big)^2 \ddot{\theta}_1
$$

$$
+ \frac{1}{2} \Bigg\{ 2J_3^{II} \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}} \tau C\theta_3 \Big) \Big[-S\alpha_{12} Ct\alpha_{23} S\theta_2 \frac{d\theta_2}{d\theta_1}
$$

$$
+ \frac{S\alpha_{34}}{S\alpha_{23}} \frac{d\tau}{d\theta_1} C\theta_3 + \frac{S\alpha_{34}}{S\alpha_{23}} \tau S\theta_3 \Big(-\tau C\alpha_{23} + C\alpha_{23} \frac{d\theta_2}{d\theta_1}
$$

$$
+ C\alpha_{12} C\alpha_{23} - S\alpha_{12} S\alpha_{23} C\theta_2 \Big) \Big] \Bigg\} \dot{\theta}_1^2 \tag{A.12k}
$$

$$
= J_3^{II} \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}} \tau C\theta_3 \Big)^2 \ddot{\theta}_1
$$

$$
+ \frac{1}{2} \Bigg\{ 2J_3^{II} \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}} \tau C\theta_3 \Big)
$$

$$
\Big(-S\alpha_{12} Ct\alpha_{23} S\theta_2 \frac{d\theta_2}{d\theta_1} + \frac{S\alpha_{34}}{S\alpha_{23}} \frac{d\tau}{d\theta_1} C\theta_3 - \frac{S\alpha_{34}}{S\alpha_{23}} \tau S\theta_3 \frac{d\theta_3}{d\theta_1} \Big) \Bigg\} \dot{\theta}_1^2 \tag{A.12l}
$$

$$
= J_3^{II} \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}} \tau C\theta_3 \Big)^2 \ddot{\theta}_1
$$

$$
+ \frac{1}{2} \Bigg\{ \frac{d}{d\theta_1} \Big(J_3^{II} \Big[S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}} \tau C\theta_3 \Big]^2 \Big) \Bigg\} \dot{\theta}_1^2 \tag{A.12m}
$$

Component J_{12}^{II}

$$
-S\alpha_{12} \Big[-J_{12}^{II} \big(\ddot{\theta}_1 S\alpha_{12} S\theta_2 C\theta_2 + 2\dot{\theta}_1 \dot{\theta}_2 S\alpha_{12} C^2\theta_2 + \dot{\theta}_1^2 S\alpha_{12} C\alpha_{12} C^2\theta_2 \big) \Big]
$$

$$
+ S\alpha_{12} \Big[-J_{12}^{II} \big(\dot{\theta}_1^2 S\alpha_{12} C\alpha_{12} S^2\theta_2 + 2\dot{\theta}_1 \dot{\theta}_2 S\alpha_{12} S^2\theta_2 - \ddot{\theta}_1 S\alpha_{12} S\theta_2 C\theta_2 \big) \Big]
$$

$$
- \Big\{ -J_{12}^{II} \big[(\dot{\theta}_1 S\alpha_{12} C\theta_2)^2 - (\dot{\theta}_1 S\alpha_{12} S\theta_2)^2 \big] B \Big\} \tag{A.13a}
$$

$$
= J_{12}^{II} S^2\alpha_{12} \Big[2\ddot{\theta}_1 S\theta_2 C\theta_2 + 2\dot{\theta}_1 \dot{\theta}_2 C^2\theta_2 - 2\dot{\theta}_1 \dot{\theta}_2 S^2\theta_2
$$

$$
+ (\dot{\theta}_1^2 C^2\theta_2 - \dot{\theta}_1^2 S^2\theta_2)(C\alpha_{12} + B) \Big] \tag{A.13b}
$$

$$
= J_{12}^{II} S^2\alpha_{12} \Big[2\ddot{\theta}_1 S\theta_2 C\theta_2 + 2\dot{\theta}_1 \dot{\theta}_2 C^2\theta_2 - 2\dot{\theta}_1 \dot{\theta}_2 S^2\theta_2
$$

$$+ (\dot{\theta}_1^2 C^2\theta_2 - \dot{\theta}_1^2 S^2\theta_2)\left(-\frac{\dot{\theta}_2}{\dot{\theta}_1}\right)\Bigg] \tag{A.13c}$$

$$=2J_{12}^{II}S^2\alpha_{12}S\theta_2 C\theta_2\ddot{\theta}_1 + J_{12}^{II}S^2\alpha_{12}(\dot{\theta}_1\dot{\theta}_2 C^2\theta_2 - \dot{\theta}_1\dot{\theta}_2 S^2\theta_2) \tag{A.13d}$$

$$=2J_{12}^{II}S^2\alpha_{12}S\theta_2 C\theta_2\ddot{\theta}_1 + \frac{1}{2}\left[2J_{12}^{II}S^2\alpha_{12}(C^2\theta_2 - S^2\theta_2)\frac{d\theta_2}{d\theta_1}\right]\dot{\theta}_1^2 \tag{A.13e}$$

$$=2J_{12}^{II}S^2\alpha_{12}S\theta_2 C\theta_2\ddot{\theta}_1 + \frac{1}{2}\left[\frac{d}{d\theta_1}\left(2J_{12}^{II}S^2\alpha_{12}S\theta_2 C\theta_2\right)\right]\dot{\theta}_1^2 \tag{A.13f}$$

Component J_{13}^{II}

$$-S\alpha_{12}\left\{-J_{13}^{II}\left[(\ddot{\theta}_1 C\alpha_{12} + \ddot{\theta}_2) - \dot{\theta}_1^2 S^2\alpha_{12}S\theta_2 C\theta_2\right]C\theta_2\right\}$$

$$+ S\alpha_{12}\left\{-J_{13}^{II}\left[(\dot{\theta}_1 C\alpha_{12} + \dot{\theta}_2)^2 - (\dot{\theta}_1 S\alpha_{12}C\theta_2)^2\right]S\theta_2\right\}$$

$$- \left[-J_{13}^{II}(-\ddot{\theta}_1 S\alpha_{12}C\theta_2 - \dot{\theta}_1^2 S\alpha_{12}C\alpha_{12}S\theta_2)\right]B \tag{A.14a}$$

$$=J_{13}^{II}S\alpha_{12}\Bigg\{\ddot{\theta}_1 C\alpha_{12} - S\alpha_{12}\Big[\ddot{\theta}_1\big(Ct\alpha_{12} + Ct\alpha_{23}C\theta_2\big)$$

$$+ \dot{\theta}_1^2 S\theta_2\big(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2\big)\Big]$$

$$- \frac{S\alpha_{34}}{S\alpha_{23}}\Big[(\dot{\tau}\dot{\theta}_1 + \tau\ddot{\theta}_1)C\theta_3 - \tau^2\dot{\theta}_1^2 C\alpha_{23}S\theta_3\Big] - \dot{\theta}_1^2 S^2\alpha_{12}S\theta_2 C\theta_2\Bigg\}C\theta_2$$

$$- J_{13}^{II}S\alpha_{12}\left[(\dot{\theta}_1 C\alpha_{12} + \dot{\theta}_2)^2 - (\dot{\theta}_1 S\alpha_{12}C\theta_2)^2\right]S\theta_2$$

$$- \left[J_{13}^{II}(\ddot{\theta}_1 S\alpha_{12}C\theta_2 + \dot{\theta}_1^2 S\alpha_{12}C\alpha_{12}S\theta_2)\right]B \tag{A.14b}$$

$$=J_{13}^{II}\Bigg\{S\alpha_{12}C\theta_2\left[C\alpha_{12} - S\alpha_{12}\big(Ct\alpha_{12} + Ct\alpha_{23}C\theta_2\big) - \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 - B\right]\ddot{\theta}_1$$

$$+ S\alpha_{12}C\theta_2\left[-S\alpha_{12}S\theta_2\big(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2\big)\right.$$

$$\left. - \frac{S\alpha_{34}}{S\alpha_{23}}\left(\frac{d\tau}{d\theta_1}C\theta_3 - \tau^2 C\alpha_{23}S\theta_3\right)\right]\dot{\theta}_1^2$$

$$- (\dot{\theta}_1 S\alpha_{12}C\theta_2)^2 S\alpha_{12}S\theta_2 - S\alpha_{12}S\theta_2(\dot{\theta}_1 C\alpha_{12} + \dot{\theta}_2)^2$$

$$- S\alpha_{12}S\theta_2(\dot{\theta}_1 S\alpha_{12}C\theta_2)^2 - \dot{\theta}_1^2 S\alpha_{12}C\alpha_{12}S\theta_2 B\Bigg\} \tag{A.14c}$$

$$=J_{13}^{II}\Bigg\{-2S\alpha_{12}C\theta_2\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)\ddot{\theta}_1$$

$$+ S\alpha_{12}C\theta_2\left[-S\alpha_{12}S\theta_2\big(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2\big)\right.$$

$$\left. - \frac{S\alpha_{34}}{S\alpha_{23}}\left(\frac{d\tau}{d\theta_1}C\theta_3 - \tau^2 C\alpha_{23}S\theta_3\right)\right]\dot{\theta}_1^2$$

$$
- S\alpha_{12}S\theta_2\big[(\dot\theta_1 C\alpha_{12} + \dot\theta_2)^2 + \dot\theta_1^2 C\alpha_{12}B\big]\bigg\} \tag{A.14d}
$$

$$
\begin{aligned}
= &- 2J_{13}^{II}S\alpha_{12}C\theta_2\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)\ddot\theta_1 \\
&+ J_{13}^{II}S\alpha_{12}\bigg\{ C\theta_2\Big[-S\alpha_{12}S\theta_2(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2) \\
&- \frac{S\alpha_{34}}{S\alpha_{23}}\left(\frac{d\tau}{d\theta_1}C\theta_3 - \tau^2 C\alpha_{23}S\theta_3\right)\Big] - S\theta_2\Big[C^2\alpha_{12} + 2C\alpha_{12}\frac{d\theta_2}{d\theta_1} \\
&+ \left(\frac{d\theta_2}{d\theta_1}\right)^2 + C\alpha_{12}\left(-C\alpha_{12} - \frac{d\theta_2}{d\theta_1}\right)\Big]\bigg\}\dot\theta_1^2
\end{aligned} \tag{A.14e}
$$

$$
\begin{aligned}
= &- 2J_{13}^{II}S\alpha_{12}C\theta_2\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)\ddot\theta_1 \\
&+ J_{13}^{II}S\alpha_{12}\bigg\{ -C\theta_2\Big[S\alpha_{12}S\theta_2(C\alpha_{12}Ct\alpha_{23} - S\alpha_{12}C\theta_2) + \frac{S\alpha_{34}}{S\alpha_{23}}\left(\frac{d\tau}{d\theta_1}C\theta_3 - \right. \\
&\left. \tau^2 C\alpha_{23}S\theta_3\right)\Big] + S\theta_2\frac{d\theta_2}{d\theta_1}\left(C\alpha_{12} + \frac{d\theta_2}{d\theta_1}\right)\bigg\}\dot\theta_1^2
\end{aligned} \tag{A.14f}
$$

$$
\begin{aligned}
= &- 2J_{13}^{II}S\alpha_{12}C\theta_2\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)\ddot\theta_1 \\
&+ \frac{1}{2}\bigg\{ 2J_{13}^{II}S\alpha_{12}\Big[S\theta_2\frac{d\theta_2}{d\theta_1}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right) \\
&- C\theta_2\frac{d}{d\theta_1}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)\Big]\bigg\}\dot\theta_1^2
\end{aligned} \tag{A.14g}
$$

$$
\begin{aligned}
= &- 2J_{13}^{II}S\alpha_{12}C\theta_2\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)\ddot\theta_1 \\
&+ \frac{1}{2}\bigg\{ \frac{d}{d\theta_1}\Big[-2J_{13}^{II}S\alpha_{12}C\theta_2\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3\right)\Big]\bigg\}\dot\theta_1^2
\end{aligned} \tag{A.14h}
$$

Component J_{23}^{II}

$$
\begin{aligned}
&-S\alpha_{12}\Big\{-J_{23}^{II}\big[-(\dot\theta_1 C\alpha_{12} + \dot\theta_2)^2 + (\dot\theta_1 S\alpha_{12}S\theta_2)^2\big]C\theta_2\Big\} \\
&+ S\alpha_{12}\Big\{-J_{23}^{II}\big[(\ddot\theta_1 C\alpha_{12} + \ddot\theta_2) + \dot\theta_1^2 S^2\alpha_{12}S\theta_2 C\theta_2\big]S\theta_2\Big\} \\
&- \big[-J_{23}^{II}(\ddot\theta_1 S\alpha_{12}S\theta_2 - \dot\theta_1^2 S\alpha_{12}C\alpha_{12}C\theta_2)\big]B
\end{aligned} \tag{A.15a}
$$

$$
\begin{aligned}
=&J_{23}\big[-S\alpha_{12}C\theta_2(\dot\theta_1 C\alpha_{12} + \dot\theta_2)^2 - S\alpha_{12}S\theta_2(\ddot\theta_1 C\alpha_{12} + \ddot\theta_2) \\
&+ \ddot\theta_1 S\alpha_{12}S\theta_2 B - \dot\theta_1^2 S\alpha_{12}C\alpha_{12}C\theta_2 B\big]
\end{aligned} \tag{A.15b}
$$

$$
= - J_{23}S\alpha_{12}S\theta_2\bigg\{\ddot\theta_1 C\alpha_{12} - S\alpha_{12}\big[\ddot\theta_1\big(Ct\alpha_{12} + Ct\alpha_{23}C\theta_2\big)
$$

$$+ \dot{\theta}_1^2 S\theta_2 \big(C\alpha_{12} Ct\alpha_{23} - S\alpha_{12} C\theta_2\big)\Big]$$

$$- \frac{S\alpha_{34}}{S\alpha_{23}}\big[(\dot{\tau}\dot{\theta}_1 + \tau\ddot{\theta}_1)C\theta_3 - \tau^2\dot{\theta}_1^2 C\alpha_{23} S\theta_3\big] - \ddot{\theta}_1 B\bigg\}$$

$$- J_{23} S\alpha_{12} C\theta_2 \big[(\dot{\theta}_1 C\alpha_{12} + \dot{\theta}_2)^2 + \dot{\theta}_1^2 C\alpha_{12} B\big] \tag{A.15c}$$

$$= - J_{23} S\alpha_{12} S\theta_2 \bigg[C\alpha_{12} - S\alpha_{12}\big(Ct\alpha_{12} + Ct\alpha_{23} C\theta_2\big) - \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 - B\bigg]\ddot{\theta}_1$$

$$- J_{23}\bigg\{ S\alpha_{12} S\theta_2 \bigg[-S\alpha_{12} S\theta_2 \big(C\alpha_{12} Ct\alpha_{23} - S\alpha_{12} C\theta_2\big)$$

$$- \frac{S\alpha_{34}}{S\alpha_{23}}\Big(\frac{d\tau}{d\theta_1} C\theta_3 - \tau^2 C\alpha_{23} S\theta_3\Big)\bigg] + S\alpha_{12} C\theta_2 \bigg[C^2\alpha_{12} + 2C\alpha_{12}\frac{d\theta_2}{d\theta_1}$$

$$+ \Big(\frac{d\theta_2}{d\theta_1}\Big)^2 + C\alpha_{12}\Big(-C\alpha_{12} - \frac{d\theta_2}{d\theta_1}\Big)\bigg]\bigg\}\dot{\theta}_1^2 \tag{A.15d}$$

$$= 2 J_{23} S\alpha_{12} S\theta_2 \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 \Big)\ddot{\theta}_1$$

$$+ J_{23}\bigg[S\alpha_{12} S\theta_2 \Big(-S\alpha_{12} Ct\alpha_{23} S\theta_2 \frac{d\theta_2}{d\theta_1} + \frac{S\alpha_{34}}{S\alpha_{23}}\frac{d\tau}{d\theta_1} C\theta_3 - \frac{S\alpha_{34}}{S\alpha_{23}}\tau S\theta_3 \frac{d\theta_3}{d\theta_1} \Big)$$

$$- S\alpha_{12} C\theta_2(-B)\bigg]\dot{\theta}_1^2 \tag{A.15e}$$

$$= 2 J_{23} S\alpha_{12} S\theta_2 \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 \Big)\ddot{\theta}_1$$

$$+ \frac{1}{2}\bigg\{ 2 J_{23} S\alpha_{12}\bigg[S\theta_2 \frac{d}{d\theta_1}\Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 \Big)$$

$$+ C\theta_2 \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 \Big)\bigg]\bigg\}\dot{\theta}_1^2 \tag{A.15f}$$

$$= 2 J_{23} S\alpha_{12} S\theta_2 \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 \Big)\ddot{\theta}_1$$

$$+ \frac{1}{2}\bigg\{ \frac{d}{d\theta_1}\bigg[2 J_{23} S\alpha_{12} S\theta_2 \Big(S\alpha_{12} Ct\alpha_{23} C\theta_2 + \frac{S\alpha_{34}}{S\alpha_{23}}\tau C\theta_3 \Big)\bigg]\bigg\}\dot{\theta}_1^2 \tag{A.15g}$$

A.3 Energy method - Lagrange's equations

According to the assumptions made, the Lagrangian $\mathcal{L}$ of the present spherical mechanism is

$$\mathcal{L} = T^I + T^{II} + T^{III} \tag{A.16}$$

where the terms in the right-hand side of Equation (A.16) represent the kinetic energies of the links constituting the mechanism. The superscript identifies the link so that ^{I} corresponds to the input link, ^{II} to the floating link and ^{III} to the output link.

Considering the angular velocities of the input, floating and output links as reported by Yang and Zhishang [4]

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot\theta_1 \end{bmatrix} \qquad \omega_2 = \begin{bmatrix} -\dot\theta_1 S\alpha_{12}C\theta_2 \\ \dot\theta_1 S\alpha_{12}S\theta_2 \\ \dot\theta_1 C\alpha_{12} + \dot\theta_2 \end{bmatrix} \qquad \omega_3 = \begin{bmatrix} 0 \\ 0 \\ \dot\theta_4 \end{bmatrix} \tag{A.17}$$

the kinetic energies of links for given reference frames become

$$T^I = \frac{1}{2}\omega_1^T J^I \omega_1 = \frac{1}{2}J_{33}^I \dot\theta_1^2 \tag{A.18}$$

$$
\begin{aligned}
T^{II} = \frac{1}{2}\omega_2^T J^{II} \omega_2 = \frac{1}{2}[& J_{11}^{II} S^2\alpha_{12}C^2\theta_2\dot\theta_1^2 \\
& + J_{22}^{II} S^2\alpha_{12}S^2\theta_2\dot\theta_1^2 \\
& + J_{33}^{II}(\dot\theta_1 C\alpha_{12} + \dot\theta_2)^2 \\
& + 2J_{12}^{II} S^2\alpha_{12}S\theta_2 C\theta_2\dot\theta_1^2 \\
& + 2J_{13}^{II}(\dot\theta_1 C\alpha_{12} + \dot\theta_2)S\alpha_{12}C\theta_2\dot\theta_1 \\
& - 2J_{23}^{II}(\dot\theta_1 C\alpha_{12} + \dot\theta_2)S\alpha_{12}S\theta_2\dot\theta_1]
\end{aligned}
\tag{A.19}
$$

$$T^{III} = \frac{1}{2}\omega_3^T J^{III} \omega_3 = \frac{1}{2}J_{33}^{III}\dot\theta_4^2 \tag{A.20}$$

with J^Q $(Q = I, II$ and $III)$ tensor of inertia.

Substituting the Equations (A.18) to (A.20) into Equation (A.16) gives

$$
\begin{aligned}
\mathcal{L} = \frac{1}{2}[& J_{33}^I \dot{\theta}_1^2 \\
& + J_{11}^{II} S^2\alpha_{12}C^2\theta_2\dot{\theta}_1^2 + J_{22}^{II} S^2\alpha_{12}S^2\theta_2\dot{\theta}_1^2 + J_{33}^{II}(\dot{\theta}_1 C\alpha_{12} + \dot{\theta}_2)^2 \\
& + 2J_{12}^{II} S^2\alpha_{12}S\theta_2 C\theta_2\dot{\theta}_1^2 + 2J_{13}^{II}(\dot{\theta}_1 C\alpha_{12} + \dot{\theta}_2)S\alpha_{12}C\theta_2\dot{\theta}_1 \\
& - 2J_{23}^{II}(\dot{\theta}_1 C\alpha_{12} + \dot{\theta}_2)S\alpha_{12}S\theta_2\dot{\theta}_1 \\
& + J_{33}^{III}\dot{\theta}_4^2]
\end{aligned}
\tag{A.21}
$$

The kinematic equations for a spherical four-link mechanism are well documented (Yang and Zhishang [4]). Therefore, let τ the transmission ratio defined by Equation (3.86), Equation (A.21) may be expressed in terms of $\dot{\theta}_1$ as follows

$$
\mathcal{L} = \frac{1}{2} I_{r1}\dot{\theta}_1^2
\tag{A.22}
$$

where

$$
\begin{aligned}
I_{r1} = & J_{33}^I + J_{11}^{II} S^2\alpha_{12}C^2\theta_2 + J_{22}^{II} S^2\alpha_{12}S^2\theta_2 \\
& + J_{33}^{II}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \tau\frac{S\alpha_{34}}{S\alpha_{23}}C\theta_3 \right)^2 + 2J_{12}^{II} S^2\alpha_{12}S\theta_2 C\theta_2 \\
& - 2J_{13}^{II}\left(S^2\alpha_{12}Ct\alpha_{23}C^2\theta_2 + \tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}C\theta_2 C\theta_3 \right) \\
& + 2J_{23}^{II}\left(S^2\alpha_{12}Ct\alpha_{23}C\theta_2 S\theta_2 + \tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}S\theta_2 C\theta_3 \right) \\
& + J_{33}^{III}\tau^2
\end{aligned}
\tag{A.23}
$$

is the equivalent inertia measured with respect to the input axis of the system, I_{r1}.

Applying the Lagrange's equations

$$
\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_r}\right) - \frac{\partial \mathcal{L}}{\partial q_r} = Q_r \qquad \text{for r=1,} \cdots \text{, N}
\tag{A.24}
$$

for generalised coordinate θ_1 yields

$$
I_{r1}\ddot{\theta}_1 + \frac{dI_{r1}}{d\theta_1}\dot{\theta}_1^2 - \frac{1}{2}\frac{dI_{r1}}{d\theta_1}\dot{\theta}_1^2 = Q_1
\tag{A.25}
$$

The generalized force Q_1 is a sum of external torque M_{E3} acting in the direction 1 and the external force M_{D3} acting in the direction 4 resolved to axes 1. Then finally

$$I_{r1}\ddot{\theta}_1 + \frac{1}{2}I'_{r1}\dot{\theta}_1^2 = M_{E3} + \tau M_{D3} \tag{A.26}$$

Equation (A.26) is the general equation of the motion resolved to the input axis of a spherical four-link mechanism, where the I'_{r1} term is

$$
\begin{aligned}
I'_{r1} = \frac{dI_{r1}}{d\theta_1} =\ & -2(J_{11}^{II} - J_{22}^{II})S^2\alpha_{12}C\theta_2 S\theta_2\frac{d\theta_2}{d\theta_1} \\
&+2J_{33}^{II}\left(S\alpha_{12}Ct\alpha_{23}C\theta_2 + \tau\frac{S\alpha_{34}}{S\alpha_{23}}C\theta_3\right)\left(-S\alpha_{12}Ct\alpha_{23}S\theta_2\frac{d\theta_2}{d\theta_1}\right. \\
&\left.+\frac{d\tau}{d\theta_1}\frac{S\alpha_{34}}{S\alpha_{23}}C\theta_3 - \tau\frac{S\alpha_{34}}{S\alpha_{23}}S\theta_3\frac{d\theta_3}{d\theta_1}\right) \\
&+2J_{12}^{II}S^2\alpha_{12}(C^2\theta_2 - S^2\theta_2) \\
&+2J_{13}^{II}\left[2S^2\alpha_{12}Ct\alpha_{23}S\theta_2 C\theta_2\frac{d\theta_2}{d\theta_1} - \frac{d\tau}{d\theta_1}\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}C\theta_2 C\theta_3\right. \\
&\left.+\tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}\left(S\theta_2 C\theta_3\frac{d\theta_2}{d\theta_1} + C\theta_2 S\theta_3\frac{d\theta_3}{d\theta_1}\right)\right] \\
&+2J_{23}^{II}\left[S^2\alpha_{12}Ct\alpha_{23}(C^2\theta_2 - S^2\theta_2) + \frac{d\tau}{d\theta_1}\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}S\theta_2 C\theta_3\right. \\
&\left.+\tau\frac{S\alpha_{12}S\alpha_{34}}{S\alpha_{23}}\left(C\theta_2 C\theta_3\frac{d\theta_2}{d\theta_1} - S\theta_2 S\theta_3\frac{d\theta_3}{d\theta_1}\right)\right] \\
&+2J_{33}^{III}\tau\frac{d\tau}{d\theta_1} \tag{A.27}
\end{aligned}
$$

Comparing the components of Equations (A.23) with (3.93) and (A.27) with (3.94), the energy method supports the result obtained with Newtonian approach.

A.4 Dynamic equation resolved to axis 4 of III-link

Substituting Equations (3.86) and (3.96) in Equation (3.95) yields

$$M_{E3} + \tau M_{D3} = I_{r1}\left(\frac{1}{\tau}\ddot{\theta}_4 - \frac{1}{\tau^2}\frac{d\tau}{d\theta_4} \cdot \dot{\theta}_4^2\right) + \frac{1}{2}I'_{r1}\frac{\dot{\theta}_4^2}{\tau^2} \tag{A.28}$$

then after a rearrangement

$$\frac{1}{\tau}M_{E3} + M_{D3} = \frac{1}{\tau^2}I_{r1}\ddot{\theta}_4 + \frac{1}{2 \cdot \tau^3}\left(\frac{d\theta_4}{d\theta_1}\frac{dI_{r1}}{d\theta_4} - 2I_{r1}\frac{d\tau}{d\theta_4}\right)\dot{\theta}_4^2 \tag{A.29}$$

Recognising the result of Equation (3.86), the square angular velocity term can be rewritten as follows

$$\frac{1}{2}\left[\frac{1}{\tau^2}\frac{dI_{r1}}{d\theta_4} + \frac{d}{d\theta_4}\left(\frac{1}{\tau^2}\right)I_{r1}\right] \tag{A.30}$$

$$\therefore \frac{1}{2}\frac{d}{d\theta_4}\left(\frac{1}{\tau^2}I_{r1}\right) \tag{A.31}$$

Using Equation (A.31) in (A.29) gives Equation (3.97).

B |

B.1   Square root of a complex number

Let $z = a + jb$, $\quad z \in \mathbb{C}$, then its square root can be found in the following manner:

$$z = a + jb = (\gamma + j\delta)^2 = (\gamma^2 - \delta^2) + 2j\gamma\delta \tag{B.1}$$

$$a = \gamma^2 - \delta^2 \qquad\qquad\qquad b = 2\gamma\delta$$

$$a^2 = \gamma^4 - 2\gamma^2\delta^2 + \delta^4 \qquad\qquad\qquad b^2 = 4\gamma^2\delta^2$$

$$a^2 + b^2 = (\gamma^2 + \delta^2)^2$$

$$[a^2 + b^2]^{\frac{1}{2}} = (\gamma^2 + \delta^2) \qquad\qquad\qquad [a^2 + b^2]^{\frac{1}{2}} = (\gamma^2 + \delta^2)$$

$$= (\gamma^2 - \delta^2) + 2\delta \qquad\qquad\qquad = -(\gamma^2 - \delta^2) + 2\gamma^2$$

$$= a + 2\delta^2 \qquad\qquad\qquad = -a + 2\gamma^2$$

$$\delta = \frac{\left[(a^2 + b^2)^{\frac{1}{2}} - a\right]^{\frac{1}{2}}}{\sqrt{2}} \qquad\qquad\qquad \gamma = \frac{\left[(a^2 + b^2)^{\frac{1}{2}} + a\right]^{\frac{1}{2}}}{\sqrt{2}}$$

$$z = a + jb = (\gamma + j\ \mathsf{sgn}(b)\ \delta)^2 \tag{B.2}$$

$$\sqrt{z} = \sqrt{a + jb} = \pm(\gamma + j\ \mathsf{sgn}(b)\ \delta) \tag{B.3}$$

C

C.1 Receptance program

The Appendix comprises five files:

- Makefile;

- function.h;

- main.c;

- receptance.c;

- function.c.

```
/****************************************************************************/
  Makefile
/****************************************************************************/

OBJ=main.c function.c receptance.c

CFLAGS= -Wall -g
LIBS=-lm -fopenmp
CC=gcc

PROG_NAME=OMP

$(PROG_NAME):      $(OBJ)
        $(CC) $(CFLAGS) -o $(PROG_NAME) $(LIBS) $(OBJ)
        rm -f *.o
        @echo " cleaned "
        @echo " "
        @echo " well done! "
        @echo " "

/****************************************************************************/
  function.h
/****************************************************************************/
/***************************
 * Receptances Method      *
 * author: carlo peressini  *
 * date: 05 july 2011      *
 * ************************/

#include <complex.h>

#define _MY_FILE_INPUT_ "input.dat"
#define _MY_FILE_LOOP_I_ "loop.dat"
#define _MY_FILE_OUTPUT_ "output.dat"
#define _MY_PI_4_ M_PI_4
#define _MY_PI_ M_PI

#define MAX_N_LOOP 5 /* Maximum number of loops present in the system */

#define max(a,b) a>b? a:b
#define min(a,b) a<b? a:b


extern int errno;
extern int tot_thread_num;

/****************************************************************************
 *                                                                         *
 *                             structures                                  *
```

```
 *                                                                             *
 *****************************************************************************/
struct SubSys{
        int type;
        int ID;
        double complex (*a11)(int, double);
        double complex (*a12)(int, double);
        double complex (*a22)(int, double);
        struct SubSys *ptr;
};

/* type_mass: only mass/inertia subsystem */
struct type_mass{
        int ID;
        double m;
        struct type_mass *ptr;
};

/* type_zero: Abutment—spring—damp—mass/inertia subsystem */
struct type_abutment{
        int ID;
        double k;
        double c;
        double m;
        struct type_abutment *ptr;
};

/* type_one: spring—damp—mass/inertia subsystem */
struct type_one{
        int ID;
        double k;
        double c;
        double m;
        struct type_one *ptr;
};

/* type_two: mass/inertia—spring—damp—mass/inertia subsystem */
struct type_two{
        int ID;
        double m1;
        double k1;
        double c1;
        double m2;
        struct type_two *ptr;
};

/* type_three: spur gear pair subsystem */
struct type_three{
        int ID;
        double I1;       /* Inertia of firt gear */
        double R1;       /* 1 gear Radius*/
        double k;        /* */
        double c;        /* */
        double I2;       /* Inertia of second gear */
        double R2;;      /* 2 gear Radius*/
        struct type_three *ptr;
};

/* type_four: continuous shaft subsystem */
struct type_four{
        int ID;
        double G;        /* Young's Modulus or Shear Modulus; S.I. unit [N/m^2] */
        double rho;      /* material density */
        double L;        /* shaft length*/
        double d;        /* shaft diameter */
        double eta;      /* hysteretic coefficient */
        struct type_four *ptr;
};

/* type_hooke 's joint */
struct type_hooke{
        int ID;
        double I1;       /* input   inertia */
        double I2;       /* output inertia */
        double phi;      /* Nutation angle, or angular displacement between axes 1 and 2 */
        double theta1;   /* fixed angular position of axes 1 for investigation */
        struct type_hooke *ptr;
};

/* co—ordinate info about loops */
struct type_loop{
        int ID;
        int LEVEL;
        int INF;
        int SUP;
        int CIN;
        int COUT;
        struct type_loop *ptr;
};

/* */
struct frank{
        int CALL;
        struct frank *ptr;
};

/* *****************************************************************************
 *                                                                             *
 *                              functions                                      *
 *                                                                             *
 *****************************************************************************/

double complex (*aux)(int, double);

void menu_function(void);
int add_element(void);
int add_type_abutment(int );
int add_type_mass(int );
int add_type_one(int );
int add_type_two(int );
int add_type_three(int );
```

```
int add_type_four(int );
int add_type_hooke(int );
void print_loop_info(void);
void print_element(void);
void print_list(struct SubSys *);
void print_subsystem(struct SubSys *); //name: print_list
void print_type_abutment(void);
void print_type_mass(void);
void print_type_one(void);
void print_type_two(void);
void print_type_three(void);
void print_type_four(void);
void print_type_hooke(void);
void print_screen_cross(int , int );
int select_subsystem(void);
int select_from_list(struct SubSys *);
int select_type_abutment_parameter(void);
int select_type_mass_parameter(void);
int select_type_one_parameter(void);
int select_type_two_parameter(void);
int select_type_three_parameter(void);
int select_type_four_parameter(void);
int select_type_hooke_parameter(void);
void Delete_All_Data(void);
struct SubSys *Remove(struct SubSys *);
int delete_element(void);
struct SubSys *del_act(struct SubSys *, int);
int delete_stored(int , int );
int delete_type_mass(int );
int delete_type_abutment(int );
int delete_type_one(int );
int delete_type_two(int );
int delete_type_three(int );
int delete_type_four(int );
int delete_type_hooke(int );
void import_data_from_file(void);
void save_config_in_file(void);

void Save_data_in_file(void);
int Menu_Save_Receptances();
void Save_F_Fixed_Recept(FILE *);
void Save_All_Recept(FILE *);

struct SubSys *Read_SubSys_List(FILE *, struct SubSys *);
struct SubSys *Find_Type_Element(int );
struct type_abutment *Find_Type_Abutment(int );
struct type_mass *Find_Type_Mass(int );
struct type_one *Find_Type_One(int );
struct type_two *Find_Type_Two(int );
struct type_three *Find_Type_Three(int );
struct type_four *Find_Type_Four(int );
struct type_hooke *Find_Type_Hooke(int );
struct SubSys *search_sys(struct SubSys *, int);
struct type_loop *Find_Type_Loop(int );

struct type_loop *Funct_Belong_to(int);

/* This  function changes the theta1 value of the arg */
int change_theta1_uj(double );

/* int ON_AIR(int , int); */
int MAX_COORD(struct SubSys *);
int min_COORD(struct SubSys *);

void Preferences(void);
void Change_Parameter(void);
struct SubSys *change_act(struct SubSys *, int );

/* Functions count elements */
int subsys_number(struct SubSys *); /* Number MACRO: only the macro subsystems */
/* Call the derivation of the receptance R=angular response point; T=torque excitatio */
void Receptance(void);
void Receptance2(void);
void mtrx_alloc(double ***, int , int );
void mtrx_free(double **, int , int );
void _c_mtrx_alloc(double complex***, int , int );
void _c_mtrx_free(double complex**, int , int );
void mtrx_screen(double ***mtrx, int r, int c);

/* New algorithm */
double complex MANAGER(struct SubSys *, double, int, int);
double complex Direct(struct SubSys *, double);
double complex CROSS(struct SubSys *, double, int, int);

/* C_Direct = Direct receptance for a Close Loop*/
double complex C_Direct(struct SubSys *, double);
double complex C_CROSS(struct SubSys *, double, int, int);

/* Deflected Shapes Functnions */
void Deflected_Shapes_Menu(void); //main function
int Deflected_Shapes(FILE *);
void John(struct type_loop *);

/* Dynamic_Torques function calculates the torque value at each co-ordinates
   over one revolution  */
void Dynamic_Torque(void);

/* Close Loop receptances */
double complex close11(int , double);
double complex close12(int , double);
double complex close22(int , double);


/*****************************************************************************
 *                                                                           *
 *                            receptances                                    *
 *                                                                           *
 *****************************************************************************/
/*
 * 0: Abutment-Spring-Damper-Mass subsystem
 * 1: Spring-Damper-Mass subsystem
```

```
* 2: Mass—Spring—Damper—Mass subsystem
* 3: Spur gear pair subsystem
* 4: Continuous shaft subsystem
* 5: Mass/Inertia subsystem
* 6: universal joint
* 67: Close loop
*
*/

/* Abutment—Spring—Damper—Mass subsystem */
double complex z11(int, double);
double complex z12(int, double);
double complex z22(int, double);

/* Spring—Damper—Mass subsystem */
double complex b11(int, double);
double complex b12(int, double);
double complex b22(int, double);

/* Mass—Spring—Damper—Mass subsystem */
double complex c11(int, double);
double complex c12(int, double);
double complex c22(int, double);

/* Spur gear pair subsystem */
double complex d11(int, double);
double complex d12(int, double);
double complex d22(int, double);

/* Continuous shaft subsystem */
double complex e11(int, double);
double complex e12(int, double);
double complex e22(int, double);

/* Mass/Inertia subsystem */
double complex m11(int, double);

/* universal joint subsystem */
double complex h11(int, double);
double complex h12(int, double);
double complex h22(int, double);

/* Close Loop */
double complex cl11(int, double);
double complex cl12(int, double);
double complex cl22(int, double);

/* Abutment—Spring—Damper—Mass subsystem */
double complex ab11(int, double);
double complex ab12(int, double);
double complex ab22(int, double);


/***************************************************************************/
 main.c
/***************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "function.h"

# ifdef _OPENMP
#   include <omp.h>
# endif

int tot_thread_num;

/***************************************************************************
 *                                                                        *
 *                                  Main                                   *
 *                                                                        *
 ***************************************************************************/

int main()
{
/*       thread_count = strtol(argv[1],NULL,10);*/
        # ifdef _OPENMP
                tot_thread_num = omp_get_num_procs();
        # else
                tot_thread_num = 1;
        # endif
        menu_function();
        return(0);
}

/***************************************************************************
 *                                                                        *
 *                             Menu_function                              *
 *                                                                        *
 ***************************************************************************/

void menu_function(void)
{
        int choice=−1;
        char pause;

        while(choice!=0){
                printf("\n\n\n\n\n\n");
                printf("\t\t********* MENU *********");
                printf("\n\n\t\tPlease, choose an option:");
                printf("\n\n\t\t1. Input new element");
                printf("\n\t\t2. Show elements");
                printf("\n\t\t4. New values of subsystem");
                printf("\n\t\t5. Import config from file");
                printf("\n\t\t6. Save config into file");
                printf("\n\t\t7. Save data into file");
                printf("\n\t\t8. Delete element");
                printf("\n\t\t9. Delete all data");
                printf("\n\t\t10. Preferences");
                printf("\n\t\t11. RECEPTANCE");
```

```
                              printf("\n\t\t13. Deflected shapes");
                              printf("\n\t\t14. Dynamic torques");
                              printf("\n\t\t0. Exit\n\n\n");
                              printf("\t\t");
                              scanf("%d",&choice);
                              printf("\n\n\n");

                                      switch(choice) {
                                              case 1:
                                                      add_element();
                                                      break;
                                              case 2:
                                                      print_element();
                                                      print_type_mass();
                                                      print_type_abutment();
                                                      print_type_one();
                                                      print_type_two();
                                                      print_type_three();
                                                      print_type_four();
                                                      print_type_hooke();
                                                      print_loop_info();
                                                      printf("\n\t\t...done");
                                                      printf("\n\n\t\tPress any key to continue...");
                                                      scanf("%c,%c",&pause,&pause);
                                                      break;
                                              case 8:
                                                      delete_element();
                                                      printf("\n\n\t\tPress any key to continue...");
                                                      scanf("%c,%c",&pause,&pause);
                                                      break;
                                              case 5:
                                                      import_data_from_file();
                                                      break;
                                              case 6:
                                                      save_config_in_file();
                                                      break;
                                              case 9:
                                                      Delete_All_Data();
                                                      break;
                                              case 4:
                                                      Change_Parameter();
                                                      break;
                                              case 7:
                                                      Save_data_in_file();
                                                      break;
                                              case 10:
                                                      Preferences();
                                                      break;
                                              case 11:
                                                      Receptance();
                                                      break;
                                              case 13:
                                                      Deflected_Shapes_Menu();
                                                      break;
                                              case 14:
                                                      Dynamic_Torque();
                                                      break;
                                      }
                      }
                      Delete_All_Data();
}

/****************************************************************************/
  receptance.c
/****************************************************************************/

#include <stdio.h>
#include <math.h>
#include <complex.h>
#include "function.h"

/****************************************************************************
 *                                                                          *
 *                              receptances                                 *
 *                                                                          *
 ****************************************************************************/

/* Abutment-Spring-Damper-Mass subsystem */

double complex z11(int id, double w)
{
        struct type_abutment *data;
        double k, c, m;

        data=Find_Type_Abutment(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return 0.;
}

double complex z12(int id, double w)
{
        struct type_abutment *data;
        double k, c, m;

        data=Find_Type_Abutment(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return 0.;
}

double complex z22(int id, double w)
{
        struct type_abutment *data;
        double k, c, m;
```

```
        data=Find_Type_Abutment(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return (1./(k+c*w*I-m*pow(w,2)));
}

/* Spring-Damper-Mass subsystem */

double complex b11(int id, double w)
{
        struct type_one *data;
        double k, c, m;

        data=Find_Type_One(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return 1./(k+c*w*I)-1./(pow(w,2)*m);
}

double complex b12(int id, double w)
{
        struct type_one *data;
        double k, c, m;

        data=Find_Type_One(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return -1./(pow(w,2)*m);
}

double complex b22(int id, double w)
{
        struct type_one *data;
        double k, c, m;

        data=Find_Type_One(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return -1./(pow(w,2)*m);
}

/* Mass-Spring-Damper-Mass subsystem */

double complex c11(int id, double w)
{
        struct type_two *data;
        double m1, k, c, m2;

        data=Find_Type_Two(id);

        m1 = data->m1;
        k = data->k1;
        c = data->c1;
        m2 = data->m2;

        return (k+c*w*I-m2*pow(w,2))/(m1*m2*pow(w,4)-(k+c*w*I)*(m1+m2)*pow(w,2));
}

double complex c12(int id, double w)
{
        struct type_two *data;
        double m1, k, c, m2;

        data=Find_Type_Two(id);

        m1 = data->m1;
        k = data->k1;
        c = data->c1;
        m2 = data->m2;

        return (k+c*w*I)/(m1*m2*pow(w,4)-(k+c*w*I)*(m1+m2)*pow(w,2));
}

double complex c22(int id, double w)
{
        struct type_two *data;
        double m1, k, c, m2;

        data=Find_Type_Two(id);

        m1 = data->m1;
        k = data->k1;
        c = data->c1;
        m2 = data->m2;

        return (k+c*w*I-m1*pow(w,2))/(m1*m2*pow(w,4)-(k+c*w*I)*(m1+m2)*pow(w,2));
}

/* Spur Gear Pair subsystem */

double complex d11(int id, double w)
{
        struct type_three *data;
        double I1, R1,  k, c, I2, R2;

        data=Find_Type_Three(id);
```

```
        I1 = data−>I1;
        R1 = data−>R1;
        k = data−>k;
        c = data−>c;
        I2 = data−>I2;
        R2 = data−>R2;

        return ((k+c*w*I)*pow(R2,2)−I2*pow(w,2))/(I1*I2*pow(w,4)−(k+c*w*I)*(pow(R2,2)*I1+pow(R1,2)*I2)*pow(w,2));
}

double complex d12(int id, double w)
{
        struct type_three *data;
        double I1, R1,  k, c, I2, R2;

        data=Find_Type_Three(id);

        I1 = data−>I1;
        R1 = data−>R1;
        k = data−>k;
        c = data−>c;
        I2 = data−>I2;
        R2 = data−>R2;

        return ((k+c*w*I)*R1*R2)/(I1*I2*pow(w,4)−(k+c*w*I)*(pow(R2,2)*I1+pow(R1,2)*I2)*pow(w,2));
}

double complex d22(int id, double w)
{
        struct type_three *data;
        double I1, R1,  k, c, I2, R2;

        data=Find_Type_Three(id);

        I1 = data−>I1;
        R1 = data−>R1;
        k = data−>k;
        c = data−>c;
        I2 = data−>I2;
        R2 = data−>R2;

        return ((k+c*w*I)*pow(R1,2)−I1*pow(w,2))/(I1*I2*pow(w,4)−(k+c*w*I)*(pow(R2,2)*I1+pow(R1,2)*I2)*pow(w,2));
}

/* Continuous shaft subsystem */

double complex e11(int id, double w)
{
        struct type_four *data;
        double G, rho, L, d, eta;        /* input parameters */
        double lambda, S, R, a, b, J;    /* auxiliary parameters */
        double complex alpha;

        data=Find_Type_Four(id);

        G = data−>G;
        rho = data−>rho;
        L = data−>L;
        d = data−>d;
        eta = data−>eta;

        lambda = w*sqrt(rho/G);

        J = (_MY_PI_/32.)*pow(d,4);
        S = − (sqrt(sqrt(1. + pow(eta,2)) − 1.))/(sqrt(2.)*sqrt(1. + pow(eta,2)));
        R = (sqrt(sqrt(1. + pow(eta,2)) + 1.))/(sqrt(2.)*sqrt(1. + pow(eta,2)));

        a = 2.*G*J*lambda*((S + eta*R)*cos(lambda*R*L)*sinh(lambda*S*L) + (eta*S − R)*sin(lambda*R*L)*cosh(lambda*S*L));
        b = 2.*G*J*lambda*((eta*S − R)*cos(lambda*R*L)*sinh(lambda*S*L) − (S + eta*R)*sin(lambda*R*L)*cosh(lambda*S*L));

        alpha =2*(a*cosh(lambda*S*L)*cos(lambda*R*L)
                − b*sinh(lambda*S*L)*sin(lambda*R*L)
                − (b*cosh(lambda*S*L)*cos(lambda*R*L)
                + a*sinh(lambda*S*L)*sin(lambda*R*L))*I)/(pow(a,2) + pow(b,2));

        return (alpha);
}

double complex e12(int id, double w)
{
        struct type_four *data;
        double G, rho, L, d, eta;        /* input parameters */
        double lambda, S, R, a, b, J;    /* auxiliary parameters */
        double complex alpha;

        data=Find_Type_Four(id);

        G = data−>G;
        rho = data−>rho;
        L = data−>L;
        d = data−>d;
        eta = data−>eta;

        lambda = w*sqrt(rho/G);

        J = (_MY_PI_/32.)*pow(d,4);
        S = − (sqrt(sqrt(1. + pow(eta,2)) − 1.))/(sqrt(2.)*sqrt(1. + pow(eta,2)));
        R = (sqrt(sqrt(1. + pow(eta,2)) + 1.))/(sqrt(2.)*sqrt(1. + pow(eta,2)));

        a = 2.*G*J*lambda*((S + eta*R)*cos(lambda*R*L)*sinh(lambda*S*L) + (eta*S − R)*sin(lambda*R*L)*cosh(lambda*S*L));
        b = 2.*G*J*lambda*((eta*S − R)*cos(lambda*R*L)*sinh(lambda*S*L) − (S + eta*R)*sin(lambda*R*L)*cosh(lambda*S*L));

        alpha =2*(a − b*I)/(pow(a,2) + pow(b,2));

        return (alpha);
}

double complex e22(int id, double w)
{
        struct type_four *data;
        double G, rho, L, d, eta;        /* input parameters */
```

```
        double lambda, S, R, a, b, J;    /* auxiliary parameters */
        double complex alpha;

        data=Find_Type_Four(id);

        G = data->G;
        rho = data->rho;
        L = data->L;
        d = data->d;
        eta = data->eta;

        lambda = w*sqrt(rho/G);

        J = (_MY_PI_/32.)*pow(d,4);
        S = -(sqrt(sqrt(1. + pow(eta,2)) - 1.))/(sqrt(2.)*sqrt(1. + pow(eta,2)));
        R = (sqrt(sqrt(1. + pow(eta,2)) + 1.))/(sqrt(2.)*sqrt(1. + pow(eta,2)));

        a = 2.*G*J*lambda*((S + eta*R)*cos(lambda*R*L)*sinh(lambda*S*L) + (eta*S - R)*sin(lambda*R*L)*cosh(lambda*S*L));
        b = 2.*G*J*lambda*((eta*S - R)*cos(lambda*R*L)*sinh(lambda*S*L) - (S + eta*R)*sin(lambda*R*L)*cosh(lambda*S*L));

        alpha =2*(a*cosh(lambda*S*L)*cos(lambda*R*L)
                - b*sinh(lambda*S*L)*sin(lambda*R*L)
                - (b*cosh(lambda*S*L)*cos(lambda*R*L)
                + a*sinh(lambda*S*L)*sin(lambda*R*L))*I)/(pow(a,2) + pow(b,2));

        return (alpha);
}

/* Mass/Inertia subsystem */

double complex m11(int id, double w)
{

        struct type_mass *data;
        double m;

        data=Find_Type_Mass(id);

        m = data->m;

        return -1/(pow(w,2)*m);
}

/* universal joint subsistem */

double complex h11(int id, double w)
{
        struct type_hooke *data;
        double I1, I2;                  /* inertia of universal s joint*/
        double phi, theta1;             /* angle of interesting: phi = nutation angle; theta1 = angle investigation */
        double tau;                     /* transmission ratio */

        data=Find_Type_Hooke(id);

        I1 = data->I1;
        I2 = data->I2;
        phi= data->phi*_MY_PI_/180;
        theta1 = data->theta1*_MY_PI_/180;

        tau = cos(phi)/(1.-pow(sin(phi),2)*pow(cos(theta1),2));

        return 1/(-pow(w,2)*(I1 + pow(tau,2)*I2));
}


double complex h12(int id, double w)
{
        struct type_hooke *data;
        double I1, I2;                  /* inertia of universal joint*/
        double phi, theta1;             /* angle of interesting: phi = nutation angle; theta1 = angle investigation */
        double tau;                     /* transmission ratio */

        data=Find_Type_Hooke(id);

        I1 = data->I1;
        I2 = data->I2;
        phi= data->phi*_MY_PI_/180;
        theta1 = data->theta1*_MY_PI_/180;

        tau = cos(phi)/(1.-pow(sin(phi),2)*pow(cos(theta1),2));

        return tau/(-pow(w,2)*(I1 + pow(tau,2)*I2));
}


double complex h22(int id, double w)
{
        struct type_hooke *data;
        double I1, I2;                  /* inertia of universal joint*/
        double phi, theta1;             /* angle of interesting: phi = nutation angle; theta1 = angle investigation */
        double tau;                     /* transmission ratio */

        data=Find_Type_Hooke(id);

        I1 = data->I1;
        I2 = data->I2;
        phi= data->phi*_MY_PI_/180;
        theta1 = data->theta1*_MY_PI_/180;

        tau = cos(phi)/(1.-pow(sin(phi),2)*pow(cos(theta1),2));

        return pow(tau,2)/(-pow(w,2)*(I1 + pow(tau,2)*I2));
}

/* Closed loop receptances */

double complex cl11(int id, double w)

{
        return (close11(id,w));
```

```
}

double complex cl12(int id, double w)
{
        return (close12(id,w));
}

double complex cl22(int id, double w)
{
        return (close22(id,w));
}


/* Abutment-Spring-Damper-Mass subsystem */

double complex az11(int id, double w)
{
        struct type_abutment *data;
        double k, c, m;

        data=Find_Type_Abutment(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return 1.;
}
double complex az12(int id, double w)
{
        struct type_abutment *data;
        double k, c, m;

        data=Find_Type_Abutment(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return ((k+c*w*I)/(k-m*pow(w,2)+c*w*I));
}
double complex az22(int id, double w)
{
        struct type_abutment *data;
        double k, c, m;

        data=Find_Type_Abutment(id);

        k = data->k;
        c = data->c;
        m = data->m;

        return (1./(k-m*pow(w,2)+c*w*I));
}
/****************************************************************************/
 function.c
/****************************************************************************/

#include <errno.h>
#include <malloc.h>
#include <float.h>
#include <math.h>
#include <complex.h>
#include <stdlib.h>
#include <time.h>
#include "function.h"


# ifdef _OPENMP
#  include <omp.h>
# endif


/* ****************************************************************************
 *                                                                          *
 *                          Global variables                                *
 *                                                                          *
 ****************************************************************************/

struct SubSys *SS_list=NULL;       /* subsystems store */

struct type_abutment *t0_list = NULL;
struct type_mass *tM_list = NULL;
struct type_one *t1_list = NULL;
struct type_two *t2_list = NULL;
struct type_three *t3_list = NULL;
struct type_four *t4_list = NULL;
struct type_hooke *th_list = NULL;
struct type_loop *L_list = NULL;          /* L_list: LOOP_list */

struct type_loop *th_info = NULL;
struct type_loop *T_info = NULL;

struct SubSys *L[MAX_N_LOOP];
struct frank *Little = NULL;

double **_DB=NULL;              /* DATABASE Matrix */
double complex **_DT=NULL;      /* DISPLACEMENTS-TORQUES Matrix */

int ID = 0;                 /* Identification number of subsystem */
int N_OF_ELM=0;             /* Total number of elements constituting the system (only input from file) */

int TOT_LOOP=0;             /* Total number of closed loop included in the system */
int N_LOOP = 0;             /* Step of uploading system from external file: */
                           /* N_LOOP addresses the storing of subsystems */
int DIS=0;                 /* co-ordinate investigated */
int FORCE=1;               /* co-ordinate subjected by the torque */
```

```
double _ROT_ANG_=0;

/*=======================================================
   OPENMULTIPROCESSOR
  =====================================================*/

int tot_thread_num;
int my_rank;

/*=======================================================
              FREQUENCY DOMAIN PARAMETERS
  =====================================================*/

double w_sup_lim = 10050.; // 1600 Hz
double delta_w = 1.;
double w_inv = 13.37; // w: investigation value

/* **************************************************************************
 *                                                                         *
 *                              add_element                                *
 *                                                                         *
 **************************************************************************/

int add_element()
{
        struct SubSys *s;
        struct type_loop *v;
        struct frank *job, *aux_hand;

        int tp;              /* tp: type of subsystem */
        int bell;            /* bell values: 1 for all lines "67 1"; 0, other cases  */
        int cl=0;            /* cl: */
        int msg = -2;

        while(msg != -1){

                printf("\t\tSelect the subsystem type:\n\n");
                printf("\t\t0  - Abutment-Spring-Damper-Inertia\n");
                printf("\t\t1  - Spring-Damper-Inertia\n");
                printf("\t\t2  - Inertia-Spring-Damper-Inertia\n");
                printf("\t\t3  - Spure gear pair\n");
                printf("\t\t4  - Continuous shaft\n");
                printf("\t\t5  - Inertia\n");
                printf("\t\t6  - Universal joint\n");
                printf("\t\t67 - Back-to-back\n");
                printf("\t\t-1 - Quit\n");
                printf("\t\t");
                scanf("%d",&tp);

                s = (struct SubSys *)malloc(sizeof(struct SubSys));     /* allocation memory */
                if(s == NULL)    printf("\n\n\t\tOut-of-Memory");        /* check the dynamic memory: NULL == out-of-memory */

                switch(tp){
                        case 0:
                                N_OF_ELM++;

                                s->type = 0;
                                s->ID = ID;
                                s->a11=z11; s->a12=z12; s->a22=z22;
                                add_type_abutment(ID);

                                break;
                        case 1:
                                N_OF_ELM++;

                                s->type = 1;
                                s->ID = ID;
                                s->a11=b11; s->a12=b12; s->a22=b22;
                                add_type_one(ID);

                                break;
                        case 2:
                                N_OF_ELM++;

                                s->type = 2;
                                s->ID = ID;
                                s->a11=c11; s->a12=c12; s->a22=c22;
                                add_type_two(ID);

                                break;
                        case 3:
                                N_OF_ELM++;

                                s->type = 3;
                                s->ID = ID;
                                s->a11=d11; s->a12=d12; s->a22=d22;
                                add_type_three(ID);
                                break;
                        case 4:
                                N_OF_ELM++;

                                s->type = 4;
                                s->ID = ID;
                                s->a11=e11; s->a12=e12; s->a22=e22;
                                add_type_four(ID);

                                break;
                        case 5:
                                N_OF_ELM++;

                                s->type = 5;
                                s->ID = ID;
                                s->a11=m11; s->a12=m11; s->a22=m11;
                                add_type_mass(ID);

                                break;
                        case 6:
                                N_OF_ELM++;
                                s->type = 6;
                                s->ID = ID;
                                s->a11=h11; s->a12=h12; s->a22=h22;
```

```
                                  add_type_hooke(ID);

                                  break;
                          case 67:
                                  printf("\n\n\t\tINFO: OPEN LOOP NUMBER: %d\n", N_LOOP);
                                  printf("\n\n\t\t== Options ==\n");
                                  printf("\t\t1 - Open;\n");
                                  printf("\t\t0 - Close;\n\n");

                                  printf("\t\t");
                                  scanf("%d",&cl);

                                  if(cl){
                                          N_LOOP++;

                                          s->type = 67;
                                          s->ID = ID;
                                          s->a11=cl11; s->a12=cl12; s->a22=cl22;
                                          bell=1;

                                          v = (struct type_loop *)malloc(sizeof(struct type_loop));

                                          printf("\t\tINPUT CO-ORDINATE: \n");
                                          printf("\t\tOUTPUT CO-ORDINATE: \n");

                                          printf("\n\t\t");
                                          scanf("%d",&v->CIN);
                                          printf("\t\t");
                                          scanf("%d",&v->COUT);

                                          v->ID = ID;
                                          v-> LEVEL = TOT_LOOP++;
                                          v->INF = ID+1;

                                          v->SUP=0;
                                          v->ptr = L_list;
                                          L_list = v;
                                          /* this is a memory: job struct stacks */

                                          job = (struct frank*)malloc(sizeof(struct frank));

                                          job->CALL = TOT_LOOP;
                                          job->ptr = Little;
                                          Little = job;

                                          ID--;
                                  }
                                  else{
                                          bell=0.;
                                          free(s);
                                          s=NULL;

                                          aux_hand = Little;
                                          Little = Little->ptr;
                                          free(aux_hand);

                                          N_LOOP--;

                                          if(!N_LOOP) v->SUP = ID;
                                          else{
                                                  v->SUP = ID;
                                                  v = v->ptr;
                                          }
                                  }
                                  break;
                          case -1:
                                          msg = -1;
                  }

                  if(msg != -1){
                          if(s!=NULL){
                                  if(!Little){
                                          s->ptr=SS_list;
                                          SS_list=s;
                                  }
                                  else if(!Little->ptr){
                                          if(bell){
                                                  s->ptr = SS_list;
                                                  SS_list = s;
                                                  bell = 0;
                                          }
                                          else{
                                                  s->ptr = L[Little->CALL - 1];
                                                  L[Little->CALL - 1] = s;
                                                  bell = 0;
                                          }
                                  }
                                  else{
                                          if(bell){
                                                  s->ptr = L[Little->ptr->CALL -  1];
                                                  L[Little->ptr->CALL - 1] = s;
                                                  bell = 0;
                                          }
                                          else{
                                                  s->ptr = L[Little->CALL -  1];
                                                  L[Little->CALL - 1] = s;
                                                  bell = 0;
                                          }
                                  }
                          }
                          ID++;
                  }
                  else free(s);
          }
  return (0);
}

int add_type_abutment(int ID)
{
          struct type_abutment *p0;
```

```
        p0=(struct type_abutment *)malloc(sizeof(struct type_abutment));

        if(p0 == NULL)
                printf("\n\n\t\tOut-of-Memory");

        p0->ID = ID;
        p0->k = 1.;
        p0->c = .01;
        p0->m = 1.1;

        p0->ptr = t0_list;
        t0_list = p0;

        return (0);

}

int add_type_mass(int ID)
{
        struct type_mass *p0;

        p0=(struct type_mass *)malloc(sizeof(struct type_mass));

        if(p0 == NULL)
                printf("\n\n\t\tOut-of-Memory");

        p0->ID = ID;
        p0->m = 1.;

        p0->ptr = tM_list;
        tM_list = p0;

        return (0);

}

int add_type_one(int ID)
{
        struct type_one *p0;

        p0=(struct type_one *)malloc(sizeof(struct type_one));

        if(p0 == NULL)
                printf("\n\n\t\tOut-of-Memory");

        p0->ID = ID;
        p0->k = 1.;
        p0->c = .01;
        p0->m = 1.1;

        p0->ptr = t1_list;
        t1_list = p0;

        return (0);

}

int add_type_two(int ID)
{
        struct type_two *p0;

        p0=(struct type_two *)malloc(sizeof(struct type_two));

        if(p0 == NULL)
                printf("\n\n\t\tOut-of-Memory");

        p0->ID = ID;
        p0->m1 = 1.2;
        p0->k1 = 1.2;
        p0->c1 = .01;
        p0->m2 = 1.2;

        p0->ptr = t2_list;
        t2_list = p0;

        return (0);

}

int add_type_three(int ID)
{
        struct type_three *p0;

        p0=(struct type_three *)malloc(sizeof(struct type_three));

        if(p0 == NULL)
                printf("\n\n\t\tOut-of-Memory");

        p0->ID = ID;
        p0->l1 = 2.3;
        p0->R1 = .4;
        p0->k = 1.2;
        p0->c = .03;
        p0->l2 = 4.3;
        p0->R2 = .3;

        p0->ptr = t3_list;
        t3_list = p0;

        return (0);

}

int add_type_four(int ID)
{
        struct type_four *p0;

        p0 = (struct type_four *)malloc(sizeof(struct type_four));

        if(p0 == NULL)
```

```
                        printf("\n\n\t\tOut−of−Memory");

        p0−>ID = ID;
        p0−>G = 80000000000.;
        p0−>rho = 7800.;
        p0−>L = 1.;
        p0−>d = .02;
        p0−>eta = .0;

        p0−>ptr = t4_list;
        t4_list = p0;

        return (0);

}


int add_type_hooke(int ID)
{
        struct type_hooke *p0;

        p0 = (struct type_hooke *)malloc(sizeof(struct type_hooke));

        if(p0 == NULL)
                printf("\n\n\t\tOut−of−Memory");

        p0−>ID = ID;
        p0−>I1 = 1.0;
        p0−>I2 = 1.0;
        p0−>phi = 30.;
        p0−>theta1 = 0.;

        p0−>ptr = th_list;
        th_list = p0;

        return (0);

}

/********************************************************************************
 *                                                                              *
 *                      compute the subsystem number                            *
 *                                                                              *
 ********************************************************************************/

/* This function counts the number of added subsystem; only MACRO subsystems  */

int subsys_number(struct SubSys *p)
{
        int count=0;

        while(p!=NULL){
                p = p−>ptr;
                count++;
        }

        return (count);
}


/********************************************************************************
 *                                                                              *
 *                      compute the MAX co−ordinate                             *
 *                                                                              *
 ********************************************************************************/

int MAX_COORD(struct SubSys *p)
{
        struct type_loop *l;

        if(p−>type != 67) return (p−>ID+1);
        else{
                l=Find_Type_Loop(p−>ID);
                return (l−>SUP+1);
        }
}

/********************************************************************************
 *                                                                              *
 *                      compute the min co−ordinate                             *
 *                                                                              *
 ********************************************************************************/

int min_COORD(struct SubSys *p)
{
        if(p){
                while(p−>ptr) {
                        p = p−>ptr;
                }
                return p−>ID;
        }
        return −1;
}

/********************************************************************************
 *                                                                              *
 *                                Print                                         *
 *                                                                              *
 ********************************************************************************/

void print_loop_info(void)
{
        struct type_loop *paux = L_list;

        printf("\n\t\tLoop Information ");
        printf("\n\n\t\tID\tLEV\tINF\tSUP\t0−PNT\t1−PNT\n");

        while(paux!=NULL){
                printf("\t\t%d\t%d\t%d\t%d\t%d\t%d\n"
                        ,paux−>ID, paux−>LEVEL, paux−>INF, paux−>SUP, paux−>CIN, paux−>COUT);
                paux=paux−>ptr;
```

```c
        }
}

void print_element(void)
{
        int i;

        printf("\t\tNumber-of-Subsystems: %d\n",N_OF_ELM);
        printf("\t\tNumber-of-Coordinates: %d\n",N_OF_ELM+TOT_LOOP);
        print_subsystem(SS_list);

        if(TOT_LOOP)
                for(i=0;i<TOT_LOOP;i++){
                        printf("\n\n\t\tClose loop %d",i);
                        print_subsystem(L[i]);
                        printf("\n\t\tNumber-of-element: %d\n",subsys_number(L[i]));
                }
}

void print_subsystem(struct SubSys *paux)
{
        printf("\n\t\tSubsystem list");
        printf("\n\n\t\tType\tID\n");

        while(paux!=NULL){
                printf("\t\t%d\t%d\n",paux->type, paux->ID);
                paux=paux->ptr;
        }
}

void print_type_abutment(void)
{
        struct type_abutment *paux=t0_list;

        printf("\n\t\tParameter type abutment list");
        printf("\n\n\t\tID\tk\tc\tm\n");

        while(paux!=NULL){
                printf("\t\t%d\t%lf\t%lf\t%lf\n",paux->ID, paux->k, paux->c, paux->m);
                paux=paux->ptr;
        }
}

void print_type_mass(void)
{
        struct type_mass *paux=tM_list;

        printf("\n\t\tParameter type Inertia list");
        printf("\n\n\t\tID\tm\n");

        while(paux!=NULL){
                printf("\t\t%d\t%lf\n",paux->ID, paux->m);
                paux=paux->ptr;
        }
}

void print_type_one(void)
{
        struct type_one *paux=t1_list;

        printf("\n\t\tParameter type one list");
        printf("\n\n\t\tID\tk\tc\tm\n");

        while(paux!=NULL){
                printf("\t\t%d\t%lf\t%lf\t%lf\n",paux->ID, paux->k, paux->c, paux->m);
                paux=paux->ptr;
        }
}

void print_type_two(void)
{
        struct type_two *paux=t2_list;

        printf("\n\t\tParameter type two list");

        printf("\n\n\t\tID\tm\tk\tc\tm\n");

        while(paux!=NULL){
                printf("\t\t%d\t%lf\t%lf\t%lf\t%lf\n"
                        ,paux->ID, paux->m1, paux->k1, paux->c1, paux->m2);
                paux=paux->ptr;
        }
}

void print_type_three(void)
{
        struct type_three *paux=t3_list;

        printf("\n\t\tParameter type three list");
        printf("\n\n\t\tID\tI1\tR1\tk\tc\tI2\tR2\n");

        while(paux!=NULL){
                printf("\t\t%d\t %lf\t %lf\t %lf\t %lf\t %lf\t %lf\n",
                        paux->ID, paux->I1, paux->R1, paux->k, paux->c, paux->I2, paux->R2);
                paux=paux->ptr;
        }
}

void print_type_four(void)
{
        struct type_four *paux = t4_list;

        printf("\n\t\tParameter type four list");
        printf("\n\n\t\tID\tG\trho\tL\td\tteta\n");

        while(paux!=NULL){
                printf("\t\t%d\t %lf\t %lf\t %lf\t %lf\t %lf\t\n"
                        ,paux->ID, paux->G, paux->rho, paux->L, paux->d, paux->eta);
                paux=paux->ptr;
        }
}
```

```c
void print_type_hooke(void)
{
        struct type_hooke *paux = th_list;

        printf("\n\t\tParameter type hook list");
        printf("\n\n\t\tID\tl1\tl2\tphi\ttheta1\n");

        while(paux!=NULL){
                printf("\t\t%d\t %lf\t %lf\t %lf\t %lf\t\n"
                        ,paux->ID, paux->l1, paux->l2, paux->phi, paux->theta1);
                paux=paux->ptr;
        }
}
/********************************************************************************
 *                                                                              *
 *                              delete_element                                  *
 *                                                                              *
 ********************************************************************************/

int delete_element(void)
{
        struct SubSys *p1=NULL;
        int id, n=0, msg;

        printf("\n\t\tRemoving element");
        printf("\n\t\t");

        id = select_from_list(SS_list);

        p1 = Find_Type_Element(id);

        if(p1!=NULL){
                n=1;
                if(p1->type!=67)        SS_list = del_act(SS_list,id);
                else {
                        printf("\n\n\t\tmsg ************");
                        printf("\n\t\tpress key");
                        printf("\n\t\t1 - delete close loop");
                        printf("\n\t\t2 - delete close loop component");
                        printf("\n\t\tQ - Quit");

                        printf("\n\n\t\t");

                        scanf("%d", &msg);

                        switch(msg){
                                case 1:
                                        SS_list = del_act(SS_list,id);
                                        while (L[0]!=NULL)      L[0] = del_act(L[0],L[0]->ID);
                                        break;
                                case 2:
                                        id = select_from_list(L[0]);
                                        L[0]=del_act(L[0], id);
                                        break;
                                default:
                                        break;
                        }
                }
        }

        if(!n)  printf("\n\t\t...element not found\n");

        return(0);
}

struct SubSys *del_act(struct SubSys *s, int id)
{
        int n = 0;
        int type;
        struct SubSys *p1=s, *p2;

        if(p1!=NULL){
                if(p1->ID == id){
                        p2 = p1;
                        s = s->ptr;
                        type = p1->type;
                        delete_stored(type, id);
                        free(p2);
                        return (s);
                }
        else
                while(p1->ptr!=NULL && n!=1){
                        if(p1->ptr->ID!=id)
                                p1=p1->ptr;
                        else {
                                n=1;
                                p2=p1->ptr;
                                p1->ptr=p1->ptr->ptr;
                                type = p2->type;
                                delete_stored(type, id);
                                free(p2);
                        }
                }
        }

        if(!n)  printf("\n\t\t...element not found\n");

        return (s);
}

int delete_stored(int type, int id)
{
        switch(type){
                case 0:
                        delete_type_abutment(id);
                        break;
                case 1:
                        delete_type_one(id);
                        break;
```

```
                case 2:
                        delete_type_two(id);
                        break;
                case 3:
                        delete_type_three(id);
                        break;
                case 4:
                        delete_type_four(id);
                        break;
                case 5:
                        delete_type_mass(id);
                        break;
                case 6:
                        delete_type_hooke(id);
                        break;
        }
        return(0);
}

int delete_type_abutment(int id)
{
        struct type_abutment *p1=t0_list, *p2;
        int n=0;

        if(p1!=NULL){
                if(p1->ID == id){
                        p2 = p1;
                        t0_list = t0_list->ptr;
                        free(p2);
                        return(0);
                }
                else {
                        while(p1->ptr!=NULL && n!=1){
                                if(p1->ptr->ID!=id)
                                        p1=p1->ptr;
                                else {
                                        n=0;
                                        p2=p1->ptr;
                                        p1->ptr=p1->ptr->ptr;
                                        free(p2);
                                        return(0);
                                }
                        }
                }
        }
        if(!n)  printf("\n\t\t...element not found\n");
        return(0);
}

int delete_type_mass(int id)
{
        struct type_mass *p1=tM_list, *p2;
        int n=0;

        if(p1!=NULL){
                if(p1->ID == id){
                        p2 = p1;
                        tM_list = tM_list->ptr;
                        free(p2);
                        return(0);
                }
                else {
                        while(p1->ptr!=NULL && n!=1){
                                if(p1->ptr->ID!=id)
                                        p1=p1->ptr;
                                else {
                                        n=0;
                                        p2=p1->ptr;
                                        p1->ptr=p1->ptr->ptr;
                                        free(p2);
                                        return(0);
                                }
                        }
                }
        }

        if(!n) {
                printf("\n\t\t...element not found\n");
        }
        return(0);
}

int delete_type_one(int id)

{
        struct type_one *p1=t1_list, *p2;
        int n=0;

        if(p1!=NULL){
                if(p1->ID == id){
                        p2 = p1;
                        t1_list = t1_list->ptr;
                        free(p2);
                        return(0);
                }
                else {
                        while(p1->ptr!=NULL && n!=1){
                                if(p1->ptr->ID!=id)
                                        p1=p1->ptr;
                                else {
                                        n=0;
                                        p2=p1->ptr;
                                        p1->ptr=p1->ptr->ptr;
                                        free(p2);
                                        return(0);
                                }
                        }
                }
        }

        if(!n) {
```

```
                              printf("\n\t\t... element not found\n");
            }
            return(0);
}

int delete_type_two(int id)
{
            struct type_two *p1=t2_list, *p2;
            int n=0;

            if(p1!=NULL){
                    if(p1->ID == id){
                            p2 = p1;
                            t2_list = t2_list->ptr;
                            free(p2);
                            return(0);
                    }
                    else {
                            while(p1->ptr!=NULL && n!=1){
                                    if(p1->ptr->ID!=id)
                                            p1=p1->ptr;
                                    else {
                                            n=0;
                                            p2=p1->ptr;
                                            p1->ptr=p1->ptr->ptr;
                                            free(p2);
                                            return(0);
                                    }
                            }
                    }
            }

            if(!n) {
                    printf("\n\t\t... element not found\n");
            }
            return(0);
}

int delete_type_three(int id)
{
            struct type_three *p1=t3_list, *p2;
            int n=0;

            if(p1!=NULL){
                    if(p1->ID == id){
                            p2 = p1;
                            t3_list = t3_list->ptr;
                            free(p2);
                            return(0);
                    }
                    else {
                            while(p1->ptr!=NULL && n!=1){
                                    if(p1->ptr->ID!=id)
                                            p1=p1->ptr;
                                    else {
                                            n=0;
                                            p2=p1->ptr;
                                            p1->ptr=p1->ptr->ptr;
                                            free(p2);
                                            return(0);
                                    }
                            }
                    }
            }

            if(!n) {
                    printf("\n\t\t... element not found\n");
            }
            return(0);
}

int delete_type_four(int id)
{
            struct type_four *p1=t4_list, *p2;
            int n=0;

            if(p1!=NULL){
                    if(p1->ID == id){
                            p2 = p1;
                            t4_list = t4_list->ptr;
                            free(p2);
                            return(0);
                    }
                    else {
                            while(p1->ptr!=NULL && n!=1){
                                    if(p1->ptr->ID!=id)
                                            p1=p1->ptr;
                                    else {
                                            n=0;
                                            p2=p1->ptr;
                                            p1->ptr=p1->ptr->ptr;
                                            free(p2);
                                            return(0);
                                    }
                            }
                    }
            }

            if(!n) {
                    printf("\n\t\t... element not found\n");
            }
            return(0);
}


int delete_type_hooke(int id)
{
            struct type_hooke *p1=th_list, *p2;
            int n=0;

            if(p1!=NULL){
```

```c
                if(p1->ID == id){
                        p2 = p1;
                        th_list = th_list->ptr;
                        free(p2);
                        return(0);
                }
                else {
                        while(p1->ptr!=NULL && n!=1){
                                if(p1->ptr->ID!=id)
                                        p1=p1->ptr;
                                else {
                                        n=0;
                                        p2=p1->ptr;
                                        p1->ptr=p1->ptr->ptr;
                                        free(p2);
                                        return(0);
                                }
                        }
                }
        }

        if(!n) {
                printf("\n\t\t...element not found\n");
        }
        return(0);
}
/*******************************************************************************
 *                                                                             *
 *                              Select element                                 *
 *                                                                             *
 *******************************************************************************/

int select_from_list(struct SubSys *s)
{
        int id;

        print_subsystem(s);

        printf("\n\n\t\tSelect ID element: ");
        printf("\n\n\t\t");

        scanf("%d", &id);

        return (id);
}

int select_subsystem(void)
{
        int id;

        printf("\n\n\t\tSelect ID element: ");
        printf("\n\n\t\t");

        scanf("%d", &id);

        return (id);
}

int select_type_abutment_parameter(void)
{
        int choice = 0;

        while (!choice){
                printf("\n\n\t\tselect the parameter\n");
                printf("\n\t\t1. Stiffness");
                printf("\n\t\t2. Damping");
                printf("\n\t\t3. Inertia");

                printf("\n\n\t\t");

                scanf("%d", &choice);
        }

        return (choice);

}

int select_type_mass_parameter(void)
{
        int choice = 0;

        while (!choice){
                printf("\n\n\t\tselect the parameter\n");
                printf("\n\t\t1. Inertia");
                printf("\n\t\t2. Abutment");

                printf("\n\n\t\t");

                scanf("%d", &choice);
        }

        return (choice);

}

int select_type_one_parameter(void)
{
        int choice = 0;

        while (!choice){
                printf("\n\n\t\tselect the parameter\n");
                printf("\n\t\t1. Stiffness");
                printf("\n\t\t2. Damping");
                printf("\n\t\t3. Inertia");

                printf("\n\n\t\t");

                scanf("%d", &choice);
        }

        return (choice);
```

```
}
int select_type_two_parameter(void)
{
        int choice = 0;

        while (!choice){
                printf("\n\n\t\tselect the parameter\n");
                printf("\n\t\t1. Inertia one");
                printf("\n\t\t2. Stiffness");
                printf("\n\t\t3. Damping");
                printf("\n\t\t4. Inertia two");

                printf("\n\n\t\t");

                scanf("%d", &choice);
        }

        return (choice);

}

int select_type_three_parameter(void)
{
        int choice = 0;

        while (!choice){
                printf("\n\n\t\tselect the parameter\n");
                printf("\n\t\t1. Inertia one");
                printf("\n\t\t2. Radius one");
                printf("\n\t\t3. Stiffness");
                printf("\n\t\t4. Damping");
                printf("\n\t\t5. Inertia two");
                printf("\n\t\t6. Radius two");

                printf("\n\n\t\t");

                scanf("%d", &choice);
        }

        return (choice);
}

int select_type_four_parameter(void)
{
        int choice = 0;

        while (!choice){
                printf("\n\n\t\tselect the parameter\n");
                printf("\n\t\t1. Shear Modulus/Young's Modulus");
                printf("\n\t\t2. Material density");
                printf("\n\t\t3. Shaft length");
                printf("\n\t\t4. Shaft diameter");
                printf("\n\t\t5. Hysteretic/Viscous damping");

                printf("\n\n\t\t");

                scanf("%d", &choice);
        }

        return (choice);
}


int select_type_hooke_parameter(void)
{
        int choice = 0;

        while (!choice){
                printf("\n\n\t\tselect the parameter\n");
                printf("\n\t\t1. Inertia 1");
                printf("\n\t\t2. Inertia 2");
                printf("\n\t\t3. Nutation angle, phi, between two axes");
                printf("\n\t\t4. Investigation angle, theta 1");

                printf("\n\n\t\t");

                scanf("%d", &choice);
        }

        return (choice);
}

/*

REAL MATRIX ALLOCATION AND FREE

*/

void mtrx_alloc(double ***m, int r, int c)
{
        int i,j;

        (*m) = (double**)malloc(r*sizeof(double*));

        for(i=0; i <= r; i++)
        {
                (*m)[i] = (double*)malloc(c*sizeof(double*));
                for(j=0; j < c; j++)
                        (*m)[i][j] = 0.;
        }
}


void mtrx_free(double **m, int r, int c)
{
        int i;

        for(i=r; i>=0 ; i--)
                free((void*) m[i]);
```

```c
        free((void*) m);
}

/*
COMPLEX MATRIX ALLOCATION AND FREE
*/

void _c_mtrx_alloc(double complex ***m, int r, int c)
{
        int p,q;

        (*m) = (double complex **)malloc(2*r*sizeof(double complex *));
        for(p=0; p<=r; p++)
        {
                (*m)[p] = (double complex *)malloc(2*c*sizeof(double complex *));
                for(q=0; q < c; q++){
                        (*m)[p][q]=0+0*I;
                }
        }
        printf("\t\t...allocation matrix....done \n");
}


void _c_mtrx_free(double complex **m, int r, int c)
{
        int i;

        for(i=r; i>=0 ; i--)
                free((void*)m[i]);
        free((void*)m);
}

void mtrx_screen(double ***mtrx, int r, int c){

        int i,j;

        for(i=0; i<= r; i++){
                for(j=0; j<c; j++)
                        printf("%f\t",(*mtrx)[i][j]);
        printf("\n");
        }
}

void Receptance(void)
{
        double start, end;
        int ch=-1, k, _SUP_;
        double w;
        double complex msg;

        _SUP_ = (int)w_sup_lim/delta_w;

        while(ch)
        {

                printf("\n\n\t\tOptions\n");
                printf("\t\t=======\n");
                printf("\t\t1 - Angular response co-ordinate\n");
                printf("\t\t2 - Torque co-ordinate\n");
                printf("\t\t3 - Both co-ordinates\n");
                printf("\t\t0 - Quit\n");

                printf("\t\t");
                scanf("%d",&ch);

                switch(ch){
                        case 3:
                                printf("\t\tNEW TORQUE: ");
                                scanf("%d",&FORCE);
                        case 1:
                                printf("\t\tNEW ANGULAR RESPONSE: ");
                                scanf("%d",&DIS);
                                break;
                        case 2:
                                printf("\t\tNEW TORQUE: ");
                                scanf("%d",&FORCE);
                                break;
                }

                if(ch){
                        printf("\t\tDIS %d\t FORCE %d\n", DIS,FORCE);
                        if(_DB){
                                mtrx_free(_DB,_SUP_,3);
                                _DB = NULL;
                        }
                        mtrx_alloc(&_DB,_SUP_,3);

                        start = omp_get_wtime();
#                       pragma omp parallel for num_threads(tot_thread_num) private(w,msg)
                        for(k=1; k<=_SUP_; k++)
                                {
                                        w = (double) k*delta_w;
                                        msg = CROSS(SS_list,w,DIS,FORCE);
                                        _DB[k][0]=w;
                                        _DB[k][1]=cabs(msg);
                                        _DB[k][2]=carg(msg);
                                }
                        end = omp_get_wtime();
                        printf("\n\t\tPROCESSING TIME %f [sec]",end-start);
                        printf("\n\n\t\t...saving data...");
                        Save_data_in_file();
                }
        }
        if(_DB){
                printf("HELLO\n");
                mtrx_free(_DB,_SUP_,3);
                _DB = NULL;
                printf("GOODBYE\n");
        }
}
```

```
void Receptance2(void)
{
        struct SubSys *p=SS_list;
        double start, end;
        int k, _SUP_;
        int _MAX_Node;
        double w;
        double complex msg;

        _SUP_ = (int)w_sup_lim/delta_w;
        _MAX_Node = MAX_COORD(p);

                for(DIS=0; DIS<=_MAX_Node; DIS++){
                        printf("\t\tDIS %d\t FORCE %d\n", DIS,FORCE);
                        if(_DB){
                                mtrx_free(_DB,_SUP_,3);
                                _DB = NULL;
                        }
                        mtrx_alloc(&_DB,_SUP_,3);

                        start = omp_get_wtime();
#                       pragma omp parallel for num_threads(tot_thread_num) private(w,msg)
                        for(k=1; k<=_SUP_; k++)
                                {
                                        w = (double) k*delta_w;
                                        msg = CROSS(SS_list,w,DIS,FORCE);
                                        _DB[k][0]=w;
                                        _DB[k][1]=cabs(msg);
                                        _DB[k][2]=carg(msg);
                                }
                        end = omp_get_wtime();
                        printf("\n\t\tPROCESSING TIME %f [sec]",end-start);
                        printf("\n\n\t\t...saving data...");
                        Save_data_in_file();
                }

        if(_DB){
                printf("HELLO\n");
                mtrx_free(_DB,_SUP_,3);
                _DB = NULL;
                printf("GOODBYE\n");
        }
}

/******************************************************************************
 *                                                                            *
 *                              find type                                     *
 *                                                                            *
 ******************************************************************************/
struct SubSys *search_sys(struct SubSys *s, int id)
{
        struct SubSys *p=s;

        if(p==NULL) return (NULL);

        while(p->ID != id && p->ptr != NULL){
                p=p->ptr;
        }
        if(p->ID==id)    return (p);
        else             return(NULL);
}

struct SubSys *Find_Type_Element(int id)
{
        struct SubSys *p=SS_list;

        if(p==NULL) return (NULL);

        while(p->ID != id && p->ptr != NULL){
                p=p->ptr;
        }
        if(p->ID==id)    return (p);
        else             return(NULL);
}

struct type_abutment *Find_Type_Abutment(int id)
{
        struct type_abutment *p=t0_list;

        while(p->ID != id){
                p=p->ptr;
        }
        return (p);
}

struct type_mass *Find_Type_Mass(int id)
{
        struct type_mass *p=tM_list;

        while(p->ID != id){
                p=p->ptr;
        }
        return (p);
}

struct type_one *Find_Type_One(int id)
{
        struct type_one *p=t1_list;

        while(p->ID != id){
                p=p->ptr;
        }
        return (p);
}

struct type_two *Find_Type_Two(int id)
{
        struct type_two *p=t2_list;
```

```
        while(p−>ID != id){
                p=p−>ptr;
        }
        return (p);
}

struct type_three *Find_Type_Three(int id)
{
        struct type_three *p=t3_list;

        while(p−>ID != id){
                p=p−>ptr;
        }
        return (p);
}

struct type_four *Find_Type_Four(int id)
{
        struct type_four *p=t4_list;

        while(p−>ID != id){
                p=p−>ptr;
        }
        return (p);
}


struct type_hooke *Find_Type_Hooke(int id)
{
        struct type_hooke *p=th_list;

        while(p−>ID != id){
                p=p−>ptr;
        }
        return (p);
}

struct type_loop *Find_Type_Loop(int id)
{
        struct type_loop *p = L_list;

        while(p−>ID != id){
                p=p−>ptr;
        }
        return (p);
}


/*******************************************************************************
 *                                                                             *
 *                     import layout from file                                 *
 *                                                                             *
 *******************************************************************************/

void import_data_from_file(void)
{
        struct SubSys *s;
        struct type_loop *v;
        struct type_abutment *t0;
        struct type_mass *tM;
        struct type_one *t1;
        struct type_two *t2;
        struct type_three *t3;
        struct type_four *t4;
        struct type_hooke *th;
        struct frank *job, *aux_hand;

        FILE *fp;

        int tp;          /* tp: type of subsystem */
        int bell;        /* bell values: 1 for all lines "67 1"; 0, other cases  */
        int cl=0;        /* cl: */

        int i;

        for(i=0;i<=MAX_N_LOOP;i++)
                L[i] = NULL;

        fp=fopen(_MY_FILE_INPUT_,"r");

        if(fp!=NULL)
        {
                while(fscanf(fp,"%d",&tp)!=EOF){

                        s = (struct SubSys *)malloc(sizeof(struct SubSys));

                        switch(tp){
                                case 0:
                                        t0 = (struct type_abutment *)malloc(sizeof(struct type_abutment));
                                        N_OF_ELM++;

                                        fscanf(fp,"%lf%lf%lf", &t0−>k, &t0−>c, &t0−>m);

                                        s−>type = tp;
                                        s−>ID = ID;
                                        s−>a11=z11; s−>a12=z12; s−>a22=z22;

                                        t0−>ID = ID;
                                        t0−>ptr = t0_list;
                                        t0_list = t0;

                                        break;
                                case 1:
                                        t1 = (struct type_one *)malloc(sizeof(struct type_one));
                                        N_OF_ELM++;

                                        fscanf(fp,"%lf%lf%lf", &t1−>k, &t1−>c, &t1−>m);

                                        s−>type = tp;
                                        s−>ID = ID;
```

```
                              s−>a11=b11; s−>a12=b12; s−>a22=b22;

                              t1−>ID = ID;
                              t1−>ptr = t1_list;
                              t1_list = t1;

                              break;
                    case 2:
                              t2 = (struct type_two *)malloc(sizeof(struct type_two));
                              N_OF_ELM++;

                              fscanf(fp,"%lf%lf%lf%lf", &t2−>m1, &t2−>k1, &t2−>c1, &t2−>m2);

                              s−>type = tp;
                              s−>ID = ID;
                              s−>a11=c11; s−>a12=c12; s−>a22=c22;

                              t2−>ID = ID;
                              t2−>ptr = t2_list;
                              t2_list = t2;

                              break;
                    case 3:
                              t3 = (struct type_three *)malloc(sizeof(struct type_three));
                              N_OF_ELM++;

                              fscanf(fp,"%lf%lf%lf%lf%lf%lf", &t3−>l1, &t3−>R1, &t3−>k, &t3−>c, &t3−>l2, &t3−>R2);

                              s−>type = tp;
                              s−>ID = ID;
                              s−>a11=d11; s−>a12=d12; s−>a22=d22;

                              t3−>ID = ID;
                              t3−>ptr = t3_list;
                              t3_list = t3;

                              break;
                    case 4:
                              t4 = (struct type_four *)malloc(sizeof(struct type_four));
                              N_OF_ELM++;

                              fscanf(fp,"%lf%lf%lf%lf%lf", &t4−>G, &t4−>rho, &t4−>L, &t4−>d, &t4−>eta);

                              s−>type = tp;
                              s−>ID = ID;
                              s−>a11=e11; s−>a12=e12; s−>a22=e22;

                              t4−>ID = ID;
                              t4−>ptr = t4_list;
                              t4_list = t4;

                              break;
                    case 5:
                              tM = (struct type_mass *)malloc(sizeof(struct type_mass));
                              N_OF_ELM++;

                              fscanf(fp,"%lf", &tM−>m);

                              s−>type = tp;
                              s−>ID = ID;
                              s−>a11=m11; s−>a12=m11; s−>a22=m11;

                              tM−>ID = ID;
                              tM−>ptr = tM_list;
                              tM_list = tM;

                              break;
                    case 6:
                              th = (struct type_hooke *)malloc(sizeof(struct type_hooke));
                              N_OF_ELM++;

                              fscanf(fp,"%lf%lf%lf%lf", &th−>l1, &th−>l2, &th−>phi, &th−>theta1);

                              s−>type = tp;
                              s−>ID = ID;
                              s−>a11=h11; s−>a12=h12; s−>a22=h22;

                              th−>ID = ID;
                              th−>ptr = th_list;
                              th_list = th;

                              break;
                    case 67:
                              fscanf(fp,"%d",&cl);
                              if(cl)
                              {
                                       N_LOOP++;

                                       s−>type = tp;
                                       s−>ID = ID;
                                       s−>a11=cl11; s−>a12=cl12; s−>a22=cl22;
                                       bell=1;

                                       v = (struct type_loop *)malloc(sizeof(struct type_loop));

                                       fscanf(fp,"%d%d",&v−>CIN,&v−>COUT);
                                       v−>ID = ID;
                                       v−> LEVEL = TOT_LOOP++;
                                       v−>INF = ID+1;

                                       v−>SUP=0;
                                       v−>ptr = L_list;
                                       L_list = v;

                                       /* this is a memory: job struct stacks */

                                       job = (struct frank *)malloc(sizeof(struct frank));

                                       job−>CALL = TOT_LOOP;
                                       job−>ptr = Little;
                                       Little = job;
```

```
                                                    ID−−;
                                                }
                                                else{

                                                    bell=0.;
                                                    free(s);
                                                    s=NULL;

                                                    aux_hand = Little;
                                                    Little = Little−>ptr;
                                                    free(aux_hand);

                                                    N_LOOP−−;

                                                    if(!N_LOOP)
                                                            v−>SUP = ID;
                                                    else{
                                                            v−>SUP = ID;
                                                            v = v−>ptr;
                                                    }
                                                }

                                                break;
                                        default:
                                                break;
                                }

                                if(s!=NULL){
                                        if(!Little){
                                                s−>ptr=SS_list;
                                                SS_list=s;
                                        }
                                        else if(!Little−>ptr){
                                                if(bell){
                                                        s−>ptr = SS_list;
                                                        SS_list = s;
                                                        bell = 0;
                                                }
                                                else{
                                                        s−>ptr = L[Little−>CALL − 1];
                                                        L[Little−>CALL − 1] = s;
                                                        bell = 0;
                                                }
                                        }
                                        else{
                                                if(bell){
                                                        s−>ptr = L[Little−>ptr−>CALL −  1];
                                                        L[Little−>ptr−>CALL − 1] = s;
                                                        bell = 0;
                                                }
                                                else{
                                                        s−>ptr = L[Little−>CALL −   1];
                                                        L[Little−>CALL − 1] = s;
                                                        bell = 0;
                                                }
                                        }
                                }

                                ID++;
                        }
                        fclose(fp);
                }
                else
                        printf("\t\t... file doesn't exist!");
}


/*******************************************************************************
 *                                                                             *
 *                       save layout into file                                 *
 *                                                                             *
 *******************************************************************************/

void save_config_in_file(void)
{
        FILE *fp;

        struct SubSys *s = SS_list;
        char pause;

        fp=fopen(_MY_FILE_INPUT_,"w");

        if(fp!=NULL)
        {
                if(s != NULL)
                        Read_SubSys_List(fp, s);
                else{
                        printf("\t\tImpossible to save NULL configuration");
                        printf("\n\n\t\t");
                        scanf("%c,%c",&pause, &pause);
                }

                fclose(fp);
        }
        else
                printf("\n\n\t\\t\tfile %s not found", _MY_FILE_INPUT_);
}

struct SubSys *Read_SubSys_List(FILE *fp,struct SubSys *s)
{
        struct type_abutment *t0;
        struct type_mass *tM;
        struct type_one *t1;
        struct type_two *t2;
        struct type_three *t3;
        struct type_four *t4;
        struct type_hooke *th;   /* type Hook's joint */

        if(s−>ptr != NULL)
                Read_SubSys_List(fp, s−>ptr);
```

```
switch(s->type){
        case 0:
                t0 = Find_Type_Abutment(s->ID);
                fprintf(fp,"%d %lf %lf %lf\n", s->type, t0->k, t0->c, t0->m);
                break;
        case 1:
                t1 = Find_Type_One(s->ID);
                fprintf(fp,"%d %lf %lf %lf\n", s->type, t1->k, t1->c, t1->m);
                break;
        case 2:
                t2 = Find_Type_Two(s->ID);
                fprintf(fp,"%d %lf %lf %lf %lf\n", s->type, t2->m1, t2->k1, t2->c1, t2->m2);
                break;
        case 3:
                t3 = Find_Type_Three(s->ID);
                fprintf(fp,"%d %lf %lf %lf %lf %lf %lf\n", s->type, t3->I1, t3->R1, t3->k, t3->c, t3->I2, t3->R2);
                break;
        case 4:
                t4 = Find_Type_Four(s->ID);
                fprintf(fp,"%d %lf %lf %lf %lf %lf\n", s->type, t4->G, t4->rho, t4->L, t4->d, t4->eta);
                break;
        case 5:
                tM = Find_Type_Mass(s->ID);
                fprintf(fp,"%d %lf\n", s->type, tM->m);
                break;
        case 6:
                th = Find_Type_Hooke(s->ID);
                fprintf(fp,"%d %lf %lf %lf %lf\n", s->type, th->I1, th->I2, th->phi, th->theta1);
                break;
        case 67:
                fprintf(fp,"%d %d\n",s->type, 1);
                Read_SubSys_List(fp, L[0]);
                fprintf(fp,"%d %d\n",s->type, 0);
                break;
        }

        return (s);
}

/*******************************************************************************
 *                                                                             *
 *                         save data into file                                 *
 *                                                                             *
 *******************************************************************************/


/* This function saves the data on a file */
void Save_data_in_file(void)
{
        int _SUP_, k;
        char file_name[10];
        FILE *fp;

        _SUP_ = (int)w_sup_lim/delta_w;
        sprintf(file_name,"RECEP_%d_%d.dat",DIS,FORCE);
        fp=fopen(file_name,"w");

        if(fp!=NULL){
                for(k=1; k <= _SUP_; k++)
                {
                        fprintf(fp,"%lf\t%e\t%e\n",_DB[k][0],_DB[k][1],_DB[k][2]);

                }
                printf("\n\t\tsave data....done\n");
        }
        else{
                perror("The following error occured:\n");
                printf("Value of errno: %d\n", errno);
        }
        fclose(fp);

}


void Save_F_Fixed_Recept(FILE * fp)
{
        int ch = -1;
        int _SUP_, k;
        double w;
        double complex msg;
        double start, end;

        _SUP_ = (int)w_sup_lim/delta_w;

        while (ch!=0) {
                printf("\n\t\tSelect\n");
                printf("\t\t=======\n");
                printf("\t\t1. to change the displacement point\n");
                printf("\t\t2. to change the loaded point\n");
                printf("\t\t0. to Quit\n");

                printf("\t\t");
                scanf("%d",&ch);

                switch(ch){
                        case 1:
                                printf("\t\tInsert theta_PNT: ");
                                scanf("%d",&DIS);
                                break;
                        case 2:
                                printf("\t\tInsert LOAD_PNT: ");
                                scanf("%d",&FORCE);
                                break;
                }

                if(ch!=0){
                        printf("\t\tDIS %d\t FORCE %d\n", DIS,FORCE);
                        start = omp_get_wtime();

#                       pragma omp parallel for num_threads(tot_thread_num) private(w)
```

```
                    for(k=1; k<=_SUP_; k++)
                        {
                                my_rank = omp_get_thread_num();

                                w = (double) k*delta_w;

                                msg = CROSS(SS_list,w,DIS,FORCE);

                                fprintf(fp,"%lf\t\t%G\t%G\t\t%d\n",w ,cabs(msg), carg(msg),my_rank);
                        }
                    end = omp_get_wtime();
                    printf("\n\t\tPROCESSING TIME %f [sec]",end-start);
                    ch = 0;
                }
        }
}


/* Function_Belong_to seeks if PNT belongs to a closed loop */
/* It returns the loop info if true, NULL if false          */
struct type_loop *Funct_Belong_to(int PNT)
{
        struct type_loop *v = L_list;
        int count = 0;

        while((v != NULL) && !count)
                {
                    if((PNT >= v->INF) && (PNT <= v->SUP)) count = 1;
                    else v=v->ptr;
                }

        return (v);
}

/*******************************************************************************
 *                                                                             *
 *        DEFLECTED SHAPES into the file: def_shapes.dat                       *
 *                                                                             *
 *******************************************************************************/

/* This function saves deflected shapes in a file called def_shapes.dat */

void Deflected_Shapes_Menu(void)
{
        FILE *fp;
        char file_name[20];

        printf("\n\n\t\tInput w investigated: ");
        scanf("%lf",&w_inv);
        printf("\n\n");

        sprintf(file_name,"DEF_SHAPES-%2lfHz.dat", w_inv);
        fp=fopen(file_name,"w");

        if(fp!=NULL){

                Deflected_Shapes(fp);
                printf("\n\t\tsave data....done\n");
        }
        else{
                perror("The following error occured:\n");
                printf("Value of errno: %d\n", errno);
        }

        fclose(fp);
}

int Deflected_Shapes(FILE *fp)
{
        struct SubSys *ELM, *p=SS_list;
        struct type_loop *l;
        int _MAX_Node;
        int _SSN; /* SubSys_list number */
        int _ANG=1, _TOR=1;
        int i, j, col=4;
        double complex ALPHA_11;
        double complex a11, a12, a22;

        _MAX_Node = MAX_COORD(p);
        _SSN = subsys_number(p);
        /* ALPHA_11 */
        ALPHA_11=CROSS(p,w_inv,_ANG,_TOR);

        if(_DT){
                _c_mtrx_free(_DT,_MAX_Node,col);/* +1 auxiliary line */
                _DT = NULL;
        }
        _c_mtrx_alloc(&_DT,_MAX_Node,col);/* +1 auxiliary line */
        /* BOUNDARY CONDITIONS */
        _DT[1][0] = 1. + 0*I;
        _DT[1][1] = _DT[1][0]/ALPHA_11;
        /* search element 0 */
        ELM = search_sys(p,0);
        a22 = ELM->a22(ELM->ID,w_inv);
        _DT[0][3]=_DT[1][0]/a22;
        /* CHECK */
        printf("_SSN: %d\n",_SSN);
        for(i=1; i<_SSN; i++){
                ELM = search_sys(p,i);
                a11 = ELM->a11(ELM->ID,w_inv);
                a12 = ELM->a12(ELM->ID,w_inv);
                a22 = ELM->a22(ELM->ID,w_inv);
                printf("a22: %lf\n",creal(a22));
                if(ELM->type != 67){
                        _DT[i][2] = _DT[i][1] - _DT[i-1][3];
                        _DT[i][3] = (_DT[i][0] - a11*_DT[i][2])/a12;
                        _DT[i+1][0] = a12*_DT[i][2] + a22*_DT[i][3];
                        printf("NO 67\n");
                }
                else{
                        printf("YES 67\n");
```

```c
                          l = Find_Type_Loop(ELM->ID);
                          _DT[i][2] = _DT[i][1] - _DT[i-1][3];
                          _DT[i][3] = (_DT[i][0] - a11*_DT[i][2])/a12;
                          _DT[l->SUP+1][0] = a12*_DT[i][2] + a22*_DT[i][3];
                          John(l);
                }
        }
        for(i=0;i<=_MAX_Node; i++){
          for(j=0;j<col;j++)
                  fprintf(fp,"%d\t%e\t%e\t",i,cabs(_DT[i][j]),carg(_DT[i][j]));
          fprintf(fp,"\n");
        }
        if(_DT){
                printf("HELLO\n");
                _c_mtrx_free(_DT,_MAX_Node,col);
                _DT = NULL;
                printf("GOODBYE\n");
        }
        return(0);
}


void John(struct type_loop *l)
{
        int _Max, _min;
        double complex b00, b01, b02, b03;
        double complex a11, a12, a22;
        struct SubSys *ELM=NULL;
        struct type_loop *m;

        _Max = MAX_COORD(L[l->LEVEL]);
        _min = min_COORD(L[l->LEVEL]);
        printf("MAX %d - min %d \n",_Max, _min);

        b00 = CROSS(L[l->LEVEL],w_inv,l->CIN,l->CIN);
        b02 = b00;
        b03 = CROSS(L[l->LEVEL],w_inv,l->CIN,_Max);
        b01 = CROSS(L[l->LEVEL],w_inv,l->CIN,l->COUT);
        /* b01 = b11 = b12 b13 */
        _DT[l->SUP][2] = (_DT[l->ID][0] - b00*_DT[l->ID][2] - b01*_DT[l->ID][3]-b03*_DT[l->SUP][1])/(b02-b03);
        _DT[l->SUP-1][3] = _DT[l->SUP][1] - _DT[l->SUP][2];
        _DT[l->SUP][0] = b00*_DT[l->ID][2] + b01*_DT[l->ID][3] + b00*_DT[l->SUP][2] + b03*_DT[l->SUP-1][3];

        ELM = L[l->LEVEL];
        while(ELM->ptr != NULL){
                a11 = ELM->a11(ELM->ID,w_inv);
                a12 = ELM->a12(ELM->ID,w_inv);
                a22 = ELM->a22(ELM->ID,w_inv);
                printf("a22: %lf\n",creal(a22));
                if(ELM->type != 3){
                        if(ELM->type != 67){
                                _DT[ELM->ID][2] = (_DT[ELM->ID+1][0] - a22*_DT[ELM->ID][3])/a12;
                                _DT[ELM->ID][0] = a11*_DT[ELM->ID][2] + a12*_DT[ELM->ID][3];
                                if(ELM->ptr->type != 67)
                                        _DT[ELM->ID-1][3] = _DT[ELM->ID][1] - _DT[ELM->ID][2];
                                else
                                        _DT[ELM->ptr->ID][3] = _DT[ELM->ID][1] - _DT[ELM->ID][2];
                                printf("NO 3\n");
                        }
                        else{
                                m = Find_Type_Loop(ELM->ID);
                                _DT[ELM->ID][2] = (_DT[m->SUP+1][0] - a22*_DT[ELM->ID][3])/a12;
                                _DT[ELM->ID][0] = a11*_DT[ELM->ID][2] + a12*_DT[ELM->ID][3];
                                if(ELM->ptr->type != 67)
                                        _DT[ELM->ID-1][3] = _DT[ELM->ID][1] - _DT[ELM->ID][2];
                                else
                                        _DT[ELM->ptr->ID][3] = _DT[ELM->ID][1] - _DT[ELM->ID][2];
                                printf("NO 3\n");
                                John(m);
                        }
                }
                else{
                        _DT[ELM->ID][2] = (_DT[ELM->ID+1][0] - a22*_DT[ELM->ID][3] - a22*_DT[l->ID][3])/a12;
                        _DT[ELM->ID][0] = a11*_DT[ELM->ID][2] + a12*_DT[ELM->ID][3] + a12*_DT[l->ID][3];
                        printf("l->ID %d\n",l->ID);
                        _DT[ELM->ID-1][3] = _DT[ELM->ID][1] - _DT[ELM->ID][2];
                        printf("YES 3\n");
                }
                ELM = ELM->ptr;
        }
}


int change_theta1_uj(double dphi)
{
        struct type_hooke *paux = th_list;

        while(paux!=NULL)
        {
                paux->theta1 +=dphi;
                paux = paux->ptr;
        }
        return 0;
}

void Dynamic_Torque(void)
{}

/****************************************************************************
 *                                                                         *
 *                         DELETE ALL                                      *
 *                                                                         *
 ****************************************************************************/

void Delete_All_Data(void)
{
        struct type_abutment *t0 = t0_list;
        struct type_mass *tM = tM_list;
        struct type_one *t1 = t1_list;
        struct type_two *t2 = t2_list;
        struct type_three *t3 = t3_list;
```

```
                struct type_four *t4 = t4_list;
                struct type_hooke *th = th_list;

                struct type_loop *v = L_list;

                int i;

                SS_list = Remove(SS_list);

                for(i=0;i<MAX_N_LOOP;i++)
                        L[i] = Remove(L[i]);

                while (t0_list != NULL){
                        t0_list = t0_list->ptr;
                        free(t0);
                        t0=t0_list;
                }

                while (t1_list != NULL){
                        t1_list = t1_list->ptr;
                        free(t1);
                        t1=t1_list;
                }

                while (t2_list != NULL){
                        t2_list = t2_list->ptr;
                        free(t2);
                        t2=t2_list;
                }

                while (t3_list != NULL){
                        t3_list = t3_list->ptr;
                        free(t3);
                        t3=t3_list;
                }

                while (t4_list != NULL){
                        t4_list = t4_list->ptr;
                        free(t4);
                        t4=t4_list;
                }

                while (tM_list != NULL){
                        tM_list = tM_list->ptr;
                        free(tM);
                        tM=tM_list;
                }

                while (th_list != NULL){
                        th_list = th_list->ptr;
                        free(th);
                        th=th_list;
                }

                while (L_list != NULL){
                        L_list = L_list->ptr;
                        free(v);
                        v = L_list;
                }

                ID = 0;
                N_OF_ELM = 0;
                TOT_LOOP = 0;
}

struct SubSys *Remove(struct SubSys *elm)
{
                struct SubSys *s = elm;

                while (elm != NULL){
                        elm = elm->ptr;
                        free(s);
                        s=elm;
                }

                return s;

}

/*****************************************************************************
 *                                                                          *
 *                           Change parameter                               *
 *                                                                          *
 *****************************************************************************/

void Change_Parameter(void)
{
                struct SubSys *s;
                int id;

                        id = select_from_list(SS_list);
                        s = search_sys(SS_list, id);
                        change_act(s, id);
}

struct SubSys *change_act(struct SubSys *s, int id)
{

                struct type_abutment *t0 = t0_list;
                struct type_mass *tM = tM_list;
                struct type_one *t1 = t1_list;
                struct type_two *t2 = t2_list;
                struct type_three *t3 = t3_list;
                struct type_four *t4 = t4_list;
                struct type_hooke *th = th_list;


                int type;
                int sel;

                type = s->type;
```

```
switch (type){
        case 0:
                t0 = Find_Type_Abutment(id);
                sel = select_type_abutment_parameter();
                printf("\n\n\t\tInput new value: ");

                        switch(sel)
                        {
                                case 1:
                                        scanf("%lf", &t0->k);
                                        break;
                                case 2:
                                        scanf("%lf", &t0->c);
                                        break;
                                case 3:
                                        scanf("%lf", &t0->m);
                                        break;
                        }

                break;

        case 1:
                t1 = Find_Type_One(id);
                sel = select_type_one_parameter();
                printf("\n\n\t\tInput new value: ");

                        switch(sel)
                        {
                                case 1:
                                        scanf("%lf", &t1->k);
                                        break;
                                case 2:
                                        scanf("%lf", &t1->c);
                                        break;
                                case 3:
                                        scanf("%lf", &t1->m);
                                        break;
                        }

                break;

        case 2:
                t2 = Find_Type_Two(id);
                sel = select_type_two_parameter();
                printf("\n\n\t\t Input new value: ");

                        switch(sel)
                        {
                                case 1:
                                        scanf("%lf", &t2->m1);
                                        break;
                                case 2:
                                        scanf("%lf", &t2->k1);
                                        break;
                                case 3:
                                        scanf("%lf", &t2->c1);
                                        break;
                                case 4:
                                        scanf("%lf", &t2->m2);
                                        break;
                        }

                break;

        case 3:
                t3 = Find_Type_Three(id);
                sel = select_type_three_parameter();
                printf("\n\n\t\t Input new value: ");

                        switch(sel)
                        {
                                case 1:
                                        scanf("%lf", &t3->l1);
                                        break;
                                case 2:
                                        scanf("%lf", &t3->R1);
                                        break;
                                case 3:
                                        scanf("%lf", &t3->k);
                                        break;
                                case 4:
                                        scanf("%lf", &t3->c);
                                        break;
                                case 5:
                                        scanf("%lf", &t3->l2);
                                        break;
                                case 6:
                                        scanf("%lf", &t3->R2);
                                        break;
                        }

                break;

        case 4:
                t4 = Find_Type_Four(id);
                sel = select_type_four_parameter();
                printf("\n\n\t\t Input new value: ");

                        switch(sel)
                        {
                                case 1:
                                        scanf("%lf", &t4->G);
                                        break;
                                case 2:
                                        scanf("%lf", &t4->rho);
                                        break;
                                case 3:
                                        scanf("%lf", &t4->L);
                                        break;
                                case 4:
```

```
                                            scanf("%lf", &t4->d);
                                            break;
                                    case 5:
                                            scanf("%lf", &t4->eta);
                                            break;
                            }

                    break;

            case 5:
                    tM = Find_Type_Mass(id);
                    sel = select_type_mass_parameter();
                    printf("\n\n\t\tInput new value: ");

                            switch(sel)
                            {
                                    case 1:
                                            scanf("%lf", &tM->m);
                                            break;
                                    case 2:
                                            tM->m = DBL_MAX;
                                            break;
                            }

                    break;

            case 6:
                    th = Find_Type_Hooke(id);
                    sel = select_type_hooke_parameter();
                    printf("\n\n\t\t Input new value:  ");

                            switch(sel)
                            {
                                    case 1:
                                            scanf("%lf", &th->l1);
                                            break;
                                    case 2:
                                            scanf("%lf", &th->l2);
                                            break;
                                    case 3:
                                            scanf("%lf", &th->phi);
                                            break;
                                    case 4:
                                            scanf("%lf", &th->theta1);
                                            break;
                            }

                    break;

            case 67:
                    id = select_from_list(L[0]);
                    s = search_sys(L[0], id);
                    change_act(s, id);
                    break;
            default:
                    break;

    }

    return s;
}


/*****************************************************************************
 *                                                                           *
 *                              Preference                                   *
 *                                                                           *
 *****************************************************************************/
void Preferences(void)
{
        int choice=-1;
        int _SUP_;

        _SUP_ = (int)w_sup_lim/delta_w;

        if(_DB){
                printf("HELLO\n");
                mtrx_free(_DB,_SUP_,3);
                _DB = NULL;
                printf("GOODBYE\n");
        }

        while(choice){

                printf("\n\n\t\tFREQUENCY DOMAIN INFORMATION:\n");
                printf("\n\n\t\tMAX angular frequency, w MAX = %lf Hz", w_sup_lim);
                printf("\n\n\t\tIncremental w value, delta_w = %lf Hz", delta_w);
                printf("\n\n\t\tSubsystem number = %d", subsys_number(SS_list));

                printf("\n\n\t\tOMP THREADS INFORMATION:\n");
                printf("\n\n\t\tNumber of threads: %d", tot_thread_num);

                printf("\n\n\t\tSelect the change:\n");
                printf("\n\n\t\t1. MAX angular frequency value");
                printf("\n\n\t\t2. Incremental value");
                printf("\n\n\t\t3. Change both values");
                printf("\n\n\t\t4. Number of threads");
                printf("\n\n\t\t0. Press to exit");

                printf("\n\n\t\t");

                scanf("%d", &choice);

                switch(choice)
                {
                        case 1:
                                printf("\n\t\tInput MAX angular frequnecy value, w_MAX = ");
                                scanf("%lf", &w_sup_lim);
                                break;
```

```
                                        case 3:
                                                printf("\n\t\tInput MAX angular frequnecy value, w_MAX = ");
                                                scanf("%lf", &w_sup_lim);
                                        case 2:
                                                printf("\n\t\tInput incremental w value, dw = ");
                                                scanf("%lf", &delta_w);
                                                break;
                                        case 4:
                                                printf("\n\t\tInput core number, ");
                                                scanf("%d", &tot_thread_num);
                                                break;
                                        default:
                                                choice = 0;
                                                break;
                        }
                }
}


/* ****************************************************************************
 *                                                                          *
 *                              CLOSE LOOP                                   *
 *                                                                          *
 ****************************************************************************/

double complex close11(int id, double w)
{
        struct type_loop *data;
        struct SubSys *pack;
        int in;

        data = Find_Type_Loop(id);
        pack = L[data->LEVEL];

        in = data->CIN;

        return C_CROSS(pack,w,in,in);
}

double complex close12(int id, double w)
{
        struct type_loop *data;
        struct SubSys *pack;
        int in, out;

        data = Find_Type_Loop(id);
        pack = L[data->LEVEL];

        in = data->CIN;
        out = data->COUT;

        return C_CROSS(pack,w,in,out);
}

double complex close22(int id, double w)
{
        struct type_loop *data;
        struct SubSys *pack;
        int out;

        data = Find_Type_Loop(id);
        pack = L[data->LEVEL];

        out = data->COUT;

        return C_CROSS(pack,w,out,out);
}

/* Close Loop functions */

double complex C_Direct(struct SubSys *s, double w)
{
        int Q;   /* Q is the last co-ordinate of the list SubSys *s  */
        int S;   /* S is the first co-ordinate of the list SubSys *s  */
        double complex Aqq, Aqs, Ass;

        Q = MAX_COORD(s);
        S = min_COORD(s);

        Aqs = CROSS(s,w,Q,S);
        Aqq = Direct(s, w);
        Ass = CROSS(s,w,S,S);

        Aqq = Aqs + (Aqq-Aqs)*(Ass-Aqs)/(Aqq-2.*Aqs+Ass);

        return (Aqq);
}

double complex C_CROSS(struct SubSys *s, double w, int i, int j)
{
        int Q;   /* Q is the last co-ordinate of the list SubSys *s  */
        int S;   /* S is the first co-ordinate of the list SubSys *s  */
        double complex Aqq, Aqs, Ass, Aij;

        Q = MAX_COORD(s);
        S = min_COORD(s);

        if((i==Q && j==S)||(i==S && j==Q))  Aij = C_Direct(s,w);
        else {
                Aqs = CROSS(s,w,Q,S);
                Aqq = Direct(s, w);
                Ass = CROSS(s,w,S,S);
                Aij = CROSS(s,w,i,j) + (CROSS(s,w,i,Q)
                      - CROSS(s,w,i,S))*(CROSS(s,w,S,j) - CROSS(s,w,Q,j))/(Aqq-2.*Aqs+Ass);
        }

        return Aij;
}

/****************************************************************************
 *                                                                          *
```

```
 *                           Direct receptance                              *
 *                                                                          *
 ****************************************************************************/

/* Functions compute the direct receptances */

double complex Direct(struct SubSys *s, double w)
{
        struct SubSys *s0 = s->ptr;
        double complex aa, ab, bb;
        double complex DRC;

        aa=s->a11(s->ID, w);
        ab=s->a12(s->ID, w);
        bb=s->a22(s->ID, w);


        if(s0 != NULL)
                DRC = bb - cpow(ab,2)/(aa + Direct(s0, w));
        else
                DRC = bb;

        return DRC;

}

/*****************************************************************************
 *                                                                          *
 *                           Cross receptance                               *
 *                                                                          *
 ****************************************************************************/

/* New CROSS manager */

double complex CROSS(struct SubSys *p, double w, int theta, int Torque)
{
        struct SubSys *p0=p->ptr;
        struct type_loop *l, *_T_info, *_th_info;
        int _MAX_Node;
        int th,T,th1;
        double complex a11;
        double complex rtn;

        _MAX_Node = MAX_COORD(p);
        th = max(theta,Torque); /* For Maxwell's theorem alpha_ij = alpha_ji so here */
        T = min(theta,Torque);  /* one uses the bottom part of the receptance matrix */
                                /* and diagonal elements                             */

        _th_info = Funct_Belong_to(th);
        _T_info = Funct_Belong_to(T);

        /* This part of function returns the receptances of a system formed by one subsystem */
        if(!p0){
                if(p->type == 67) l = Find_Type_Loop(p->ID);

                if(T == p->ID){
                        if(th  == T) rtn = p->a11(p->ID,w);
                        else if (th == _MAX_Node) rtn = p->a12(p->ID,w);
                        else rtn = C_CROSS(L[l->LEVEL],w,th,T);
                }
                else if((T > p->ID) && (T < _MAX_Node)){
                        if(th < _MAX_Node) rtn = C_CROSS(L[l->LEVEL],w,th,T);
                        else {
                        th = l->COUT;
                        rtn = C_CROSS(L[l->LEVEL],w,th,T);
                        }
                }
                else if (T == _MAX_Node && th == T ) rtn = p->a22(p->ID,w);
        }

        else{
                if(p->type == 67) l = Find_Type_Loop(p->ID); /* There could be a problem of segmentation default */

                if(th == _MAX_Node){
                        if(T == th) rtn = Direct(p,w);
                        else if((T > p->ID) && (T < _MAX_Node)){
                                th1 = l->COUT;
                                rtn = C_CROSS(L[l->LEVEL],w,th1,T)
                                    - C_CROSS(L[l->LEVEL],w,th1,p->ID)
                                    *C_CROSS(L[l->LEVEL],w,T,p->ID)
                                    /(C_CROSS(L[l->LEVEL],w,p->ID,p->ID) + CROSS(p0,w,p->ID,p->ID));
                        }
                        else rtn = p->a12(p->ID,w)*CROSS(p0,w,p->ID,T)/(p->a11(p->ID,w)+CROSS(p0,w,p->ID,p->ID));
                }
                else if((th > p->ID) && (th < _MAX_Node)){
                        if((T > p->ID) && (T < _MAX_Node))
                                rtn = C_CROSS(L[l->LEVEL],w,th,T)
                                    - C_CROSS(L[l->LEVEL],w,th,p->ID)
                                    *C_CROSS(L[l->LEVEL],w,T,p->ID)
                                    /(C_CROSS(L[l->LEVEL],w,p->ID,p->ID) + CROSS(p0,w,p->ID,p->ID));
                        else rtn = C_CROSS(L[l->LEVEL],w,th,p->ID)*CROSS(p0,w,T,p->ID)
                                    /(C_CROSS(L[l->LEVEL],w,p->ID,p->ID) + CROSS(p0,w,p->ID,p->ID));
                }
                else if(th == p->ID){
                        a11=p->a11(p->ID,w);
                        rtn = a11*CROSS(p0,w,p->ID,T)/(a11 + CROSS(p0,w,p->ID,p->ID));
                }
                else rtn = CROSS(p0,w,th,T) - CROSS(p0,w,th,p->ID)
                            *CROSS(p0,w,p->ID,T)/(p->a11(p->ID,w)
                            + CROSS(p0,w,p->ID,p->ID));
        }
        return rtn;
}
```

Publications from PhD Candidature

At the time of thesis publication.

Conference paper:

C. Peressini, A.L. Guzzomi, D.C. Hesterman, Torsional Receptances and Variable Inertia of a Two-Inertia Model of a Universal Joint, EUCOMES 2012, Santander, Spain.

Journal paper:

A.L. Guzzomi, C. Peressini, D.C. Hesterman, Novel passive device to balance inertia variation in engines, under review (2012): Mechanism and Machine Theory.

References

[1] Guzzomi A.L., Molari P.G. Sulla vita a fatica di alberi scanalati. In *AIAS 2008 Atti del XXXVII Convegno Nazionale Roma, 10-13 September*, 2008.

[2] Guzzomi A.L. , Maraldi M., Molari P.G. Concept Design of a Novel Back-to-Back Mechanically Passive Torsional Fatigue Test Rig. *unpulished internal report*, 2010.

[3] Drew S.J., Hesterman D.C., Stone B.J. The Torsional Excitation of Variable Inertia Effects in a Reciprocating Engine. *Mechanical Systems and Signal Processing*, 13:125–144, 1999.

[4] Yang A.T., Zhishang S. Dynamic Force and Torque Analysis of Spherical Four-Link Mechanisms. *ASME J. Mechanisms, Transmissions, and Automation in Design*, 105:492–497, September 1983.

[5] Leishman B.A., Drew S.J., Stone B.J. Torsional vibration of a back-to-back gearbox rig. Part 1: frequency domain modal analysis. *Proc. IMechE, Part K: Journal of Multi-body Dynamics*, 214(3):143–162, 2000.

[6] Wöhler A. Wöhler's experiment on the strength of metals. *Engineering*, 4:160–161, 1867.

[7] Albert, W.A.J. Über Treibseile am Herz. Archiv für Mineralogie, Georgnosie. *Bergbau und Hüttenkunde*, 10:215 – 234, 1837.

[8] Schutz, W. A history of fatigue. *Engineering Fracture Mechanics*, 54(2):263 – 300, 1996.

[9] Drew S.J., Stone B.J. Torsional damping of a back-to-back gearbox rig: experimental measurements and frequency domain modelling. *Proc. IMechE, Part K: Journal of Multi-body Dynamics*, 216(2):157–168, June 2002.

[10] Deans W.F. Foong C.H., Wiercigroch M. An experimental rig to investigate fatigue crack growth under dynamic loading. *Meccanica*, 38:19 – 31, 2003.

[11] Deans W.F. Foong C.H., Wiercigroch M. Novel dynamic fatigue-testing device: design and measurements. *Measurement science and technology*, 17:2218 – 2226, 2006.

[12] Wiercigroch M. Boltežar M. Jakšić N., Foong C.H. Parameter identification of the fatigue-testing rig. *International Journal of Mechanical Sciences*, 50(7):1142 – 1152, 2008.

[13] Mayer H. Ultrasonic torsion and tension-compression fatigue testing: Measuring principles and investigations on 2024-t351 aluminium alloy. *International journal of fatigue*, pages 1446–1455, 2006.

[14] Issler L. *Festigkeitsverhalten metallischer Werkstoffe bei mehrachsiger phasenverschobener Schwingbeanspruchung*. PhD thesis, University of Stuttgart, Germany., 1973.

[15] Lempp W. *Festigkeitsverhalten von Stählen bei mehrachsiger Dauerschwingbeanspruchung durch Normalapannungen mit überlagerten phasengleichen und phasenverschobener Schubspannungen*. PhD thesis, University of Stuttgart, Germany., 1977.

[16] Marines-Garcia I., Doucet J-P.,Bathias C. Development of a new device to perform torsional ultrasonic fatigue testing. *International journal of fatigue*, 29:2094 – 2101, 2007.

[17] Tschegg E. Stanzl-Tschegg S.E., Mayer H.R. High frequency method for torsion fatigue testing. *Ultrasonic*, 31(4):275 – 280, 1993.

[18] Mayer H. Fatigue lifetime investigations in the very high cycle regime using the ultrasonic resonance testing methode. In *3rd International conference on very high cycle fatigue (VHCF-3)*, Shiga and Kyoto, Japan, September.

[19] Nerantzis I. Mihailidis A. A new system for testing gear under variable torque and speed. *Recent Pat Mech Eng*, 2:179 – 192, 2009.

[20] Sargeant M.A., Drew S.J., Stone B.J. Coupled torsional and transverse vibration of a back-to-back gearbox rig. *Proc. IMechE, Part K: Journal of Multi-body Dynamics*, 219:259–273, 2005.

[21] Höhn B.R. Laboratories at work: institute for machine elements, gear research centre (fzg). *Tribotest*, 3(3):325 – 340, 1997.

[22] Fischer I.S., Paul R.N. Kinematic Displacement Analysis of a Double-Cardan-Joint Driveline. *Journal of Mechanical Design*, 113:263–271, 1991.

[23] Mazzei Jr A.J., Argento A., Scott R.A. Dynamic Stability of a Rotating Shaft Driven Through a Universal Joint. *Journal of Sound and Vibration*, 222(1):19 – 47, 1999.

[24] Lee A.C Sheu P.P, Chieng W.H. Modeling and analysis of the intermediate shaft between two universal joints. *Trans. ASME J. Vibration and Acoustics*, 118:88 − 99, 1996.

[25] Porat I. Moment Transmission by a Universal Joint. *J. Mechanisms and Machine Theory*, 15(4):245–254, 1980.

[26] Fischer I.S., Freudenstein F. Internal Force and Moment Transmission in a Cardan Joint With Manufacturing Tolerances. *ASME J. Mechanisms, Transmissions, and Automation in Design*, 106:301–311, December 1984.

[27] Yang A.T. Inertia Force Analysis of Spatial Mechanisms. *ASME J. Engineering for Industry*, pages 27–32, February 1971.

[28] Sextus Empiricus. *Adversus Mathematicos*, volume VII.

[29] Howard I.M., Stone B.J. A Receptance Technique for the Modelling of the Vibration Characteristics of any Beam-Type Structure. *Annals of the CIRP*, 37(1):355 − 360, 1988.

[30] Bishop R.E.D. and Johnson D.C. *The Mechanics of Vibration*. Cambridge at the University Press, 1960.

[31] Li H.,Hesterman D.C., Stone B.J. A System Approach for Modelling in the Time Domain. In *International Mechanical Engineering congress, Perth, Western Australia*, pages 13 − 18, May 15 - 19 1994.

[32] Li H. *The Modelling and Excitation of Complex Torsional Systems*. PhD thesis, Department of Mechanical and Materials Engineering, University of Western Australia, Perth, Western Australia, 1996.

[33] Voormeeren S.N. de Klerk D., Rixen D.J. General Framework for Dynamic Substructuring: History, Review, and Classification of Techniques. *AIAA Journal*, 46(5):1169 − 1181, 2008.

[34] Timoshenko S. and Goodier J.N. *Theory of elasticity*. McGraw-Hill, 1969.

[35] Litvin F.L. and Fuentes A. *Gear Geometry and Applied Theory*. Cambridge at the University Press, 2nd edition, 2004.

[36] Thomson W.T. *Theory of Vibration with applications*. Stanley Thornes (Publishers) Ltd, 4th edition, 1993.

[37] Kernighan B.W.,Ritchie D.M. *The C programming language*. Prentice Hall, 1978.

[38] Van Der Linden P. *Expert C Programming: Deep C Secrets*. Prentice Hall Professional, 1994.

[39] Reuleaux F. *The constructor: A hand-book of Machine Design* . Philadelphia: H.H. Suplee, 1893.

[40] Derry S., Stone B.J. The torsional receptances in closed form of continuous bar with distributed damping. *in preparetion*.

[41] Peressini C., Guzzomi A.L., Hesterman D.C. Torsional receptances and variable inertia of a two-inertia model of a universal joint. In *4th European Conference on Mechanism Science*, Santander, Spain, September 2012.

[42] Derry S., Drew S.J., Stone B.J. The Torsional Vibration of a Damped Continuous Bar. *In Proceedings of 8th International Congress on* Experimental Mechanics*, Nashville, Tennesse*, pages 72–73, 1996.

[43] Neumark S., Techn.Sc.D., F.R.Ae.S. Concept of Complex StiffnessApplied to Problems of Oscillations with Viscous and Hysteretic Damping. Technical report, Aeronautical Research Council, 1962.

[44] O'Neil P.V. *Advanced engineering mathematics*. PWS-Kent Pub. Co., c1995., 4th edition, 1995.

[45] Seherr-Thoss H.C., Schmelz F., Aucktor E. *Universal joints and driveshafts: analysis, design, applications*. Springer, 2nd edition, 2006.

[46] Mills A. Robert Hooke's universal joint and its application to sundials and the sundial-clock. *Notes Rec. R. Soc.*, 61:219–236, 2007.

[47] Russo L. *The Forgotten Revolution. How Science Was Born in 300 BC and Why It Had Be Reborn*. Springer, 1996.

[48] Leishman B.A., Drew S.J., Stone B.J. Torsional vibration of a back-to-back gearbox rig. Part 2: time domain modelling and verification. *Proc. IMechE, Part K: Journal of Multi-body Dynamics*, 214(3):163–179, 2000.

[49] Guzzomi A.L., Hesterman D.C., Stone B.J. Variable inertia effects of an engine including piston friction and a crank or gudgeon pin offset. *Proc. IMechE, Part D: Journal of Automobile Eng.*, 222:397–414, 2008.

[50] Guzzomi A.L. *Torsional vibration of powertrains: an investigation of some common assumptions*. VDM Verlag Dr. Müller, 2007.

[51] Hesterman D.C. *Torsional Vibration in Reciprocating Pumps and Engines.* PhD thesis, Department of Mechanical Engineering, University of Western Australia, Perth, Western Australia, 1992.

[52] Drew S.J., Stone B.J. Torsional (Rotational) Vibration: Excitation of Small Rotating Machines. *Journal of Sound and Vibration*, 201(4):437–463, 1997.

[53] Guzzomi A.L., Hesterman D.C., Stone B.J. Some Effects of Piston Friction and Crank or Gudgeon Pin Offset on Crankshaft Torsional Vibration. *Journal of Ship Research*, 54(1):41–52, 2010.

[54] Guzzomi A.L., Peressini C., Hesterman D.C. Novel passive device to balance inertia variation in engines. under review 2012.

[55] Yang A.T., Freudenstein F. Application of Dual-Number Quaternion Algebra to the Analysis of Spatial Mechanism. *ASME J. Applied Mechanics*, pages 300–308, June 1964.

[56] Yang A.T. Acceleration Analysis of Spatial Four-Link Mechanisms. *ASME J. Engineering for Industry*, pages 296–300, August 1966.

[57] Chen C.K., Freudenstein F. Dynamic Analysis of a Universal Joint With Manufacturing Tolerances. *ASME J. Mechanisms, Transmissions, and Automation in Design*, 108:524–532, December 1986.

[58] Stefanelli R. Pennestrì E. Linear algebra and numerical algorithms using dual numbers, 2007.

[59] Brand L. *Vector and Tensor Analysis*. john Wiley and Sons, Inc., New York, N.Y., 6th edition, 1957.

[60] Kato M., Ota H. Lateral Excitation of a Rotating Shaft Driven by a Universal Joint with Friction. *ASME Journal of Vibration and Acoustics*, 112:298 – 303, 1990.

[61] Yoon K.Y., Rao S.S. Dynamic load analysis of spur gear using a new tooth profile. *Trans. ASME, J. Mech. Design*, 118:1 – 6, 1996.

[62] Drew S.J., Stone B.J. Torsional damping measurements for a gearbox. *Mechanical System and Signal Processing*, 19:1096–1106, 2005.