NLR TP 97379 U

NATIONAAL LUCHT- EN RUIMTEVAARTLABORATORIUM

NATIONAL AEROSPACE LABORATORY NLR

THE NETHERLANDS

NLR TP 97379 U

A MATLAB PROGRAM TO STUDY GUST LOADING ON A SIMPLE AIRCRAFT MODEL

by

W.J. Vink and J.B. de Jonge

NLR TECHNICAL PUBLICATION

TP 97379 U

A MATLAB PROGRAM TO STUDY GUST LOADING ON A SIMPLE AIRCRAFT MODEL

by

W.J. Vink and J.B. de Jonge

This report has been prepared under a contract awarded by the Netherlands Department of Civil Aviation, contract number RB-RLD(LI&VI) 1997: 1.1.2.

Division : Structures and Materials
Prepared : WJV/ 31/10 JBdJ/ 31/10
Approved : HHO/ 31/10

Completed : 970729
Order number : 106.663
Typ. : JvE

Summary

The present report is part of a "Manual" on aircraft loads, that is being prepared by NLR. It contains a computer program, to be used in combination with "MATLAB4 for Windows" (Student Edition) software, to study aircraft loading due to turbulence. It includes a simple aircraft response model with two rigid and three flexible symmetric degrees of freedom and allows calculation of various structural responses due to discrete (1-cos) gusts as well as continuous turbulence. The program is intended as a "tool" which will also be extensively used in a next report to be prepared, dealing with gust loads and requirements.

Contents

(83 pages in total)

List of symbols

A	system matrix as function of frequency	
b	wing span	[m]
bt	horizontal tail span	[m]
C	state-space matrix relating outputs to generalized coordinates	
$\bar{c}$	mean aerodynamic chord	[m]
cw	local chord of a wing strip	[m]
ct	local chord of a tail strip	[m]
C_{z_α}	local aerodynamic z-force coefficient for wing or tail	
D	state-space matrix relating outputs to inputs (gust)	
	generalized damping matrix	[kg/s]
EI	bending stiffness	[Nm^2]
ew	distance between wing leading edge and elastic axis in chords	
et	distance between tail leading edge and elastic axis in tail chords	
$\bar{F}$	generalized force vector	[kg s^{-2}]
	aerodynamic force, moment	[N], [Nm]
GJ	torsional stiffness	[Nm^2]
H	transfer function (function of frequency)	
h	impulse response function (function of time)	
I_{xl}	moment of inertia around local x-axis	[kg m^2]
I_y	aircraft moment of inertia around lateral axis	[kg m^2]
I_{yl}	moment of inertia around local y-axis	[kg m^2]
j	$j^2 = -1$	
K	generalized stiffness matrix	[kg s^{-2}]
$[k_b]_k$, $[k_t]_k$	stiffness matrices of finite element k for bending and torsion	[Nm^3]
L	lift force	[N]
	turbulence scale length	[m]
lt	distance between stabilizer elastic axis and aircraft centre of gravity	[m]
l_t	distance between second wing strip and tail	[m]
M	generalized mass matrix	[kg]
	a moment	[Nm]
m	aircraft mass	[kg]
$[m]_k$	mass matrix of finite element k	
m_l	mass of an element	[kg]
N(0)	number of positive zero-level crossings	
n_z	load factor	

Q_r	generalized aerodynamic matrix for motion response	[kg s^{-2}]
Q_w	generalized aerodynamic matrix for gust input	[kg/s]
q	pitch velocity	[rad/s]
$\bar{r}$	location vector	[m]
T	kinetic energy	[Nm]
t	time	[s]
U	elastic energy	[Nm]
V	velocity	[m/s]
	aircraft velocity	[m/s]
w	displacement in z-direction	[m]
$\bar{w}$	displacement vector	[m]
w_g	gust input signal (function of time or frequency)	[m/s], [m]
x	x-coordinate	
xl	local x-coordinate	
xt	distance between stabilizer elastic axis and axis system origin	[m]
y	an output quantity of the aircraft model	
	y-coordinate	
z	z-coordinate	
	z-displacement of the 3/4 chord point of a strip	[m]
α	angle of incidence	[rad]
Δ	increment	
δ	Dirac function	
ε	downwash angle	[rad]
θ	pitch rotation angle	[rad]
$\theta_{i_{elastic}}$	pitch rotation angle of a strip due to elastic deformations only	[rad]
ψ	torsional rotation angle in a beam element	[rad]
Λ	sweepback angle	[rad]
ξ	generalized coordinate	
ρ	air density	[kg m^{-3}]
	correlation coefficient	
σ	standard deviation	
τ	time delay	[s]
$\Phi_w{}^n$	normalized von Karman power spectral density	[s]
$\bar{\phi}$	displacement mode shape	
ϕ_{sear}	Sears function	
ϕ_{theo}	Theodorsen function	
φ	phase angle	[rad]
ω	radial frequency	[rad/s]
ω_y	rotational velocity around y-axis of a moving axis system	[rad/s]

1 Introduction

The Netherlands Department of Civil Aviation (RLD) has contracted NLR to prepare a "Manual" on aircraft loads. This Manual, which will be published in separate volumes, is intended for Engineers and Technicians of the Airworthiness branch as well as in the industry dealing with structural loads analysis and certification. Apart from providing an overview of the physics of aircraft loading and the way loads are being dealt with in current airworthiness requirements, the Manual should provide "tools" to assess the effects of changes in aircraft design or changes in requirements on structural design loads.

A first volume dealing with aircraft loads in pitching manoeuvres, intended as "pilot", has been published previously (Ref. 1).

The present report is the first part of a volume dealing with gust loads on aircraft. It presents the development and a complete description of a computer program to study (symmetrical) gust loads on aircraft. This program must be used in combination with the software package (the student edition of) MATLAB4 for Windows. It includes a simple symmetrical aircraft model with 5 degrees of freedom and allows calculation of a number of structural load quantities due to discrete gusts as well as continuous turbulence.
All response parameters can be varied in order to study their effect on the magnitude of gust-induced loads.
This computer program will be extensively used in an other report to be prepared, which will study gust loads on aircraft, with specific reference to the current Airworthiness Requirements.

Chapter 2 gives a general overview of the aircraft response analysis procedures underlying the computer program.

The development of this program, including a full description of the aircraft response model, is presented in chapter 3.

Chapter 4 presents some guidance instructions on how to use the program.

A complete listing of all matlab-routines (M-files) is given in appendix B. These M-files, constituting the computer program, are also included on the floppy disk going with this report.

2 General overview of aircraft response analysis procedures

2.1 The transfer function concept

The aircraft response due to turbulence is schematically illustrated in the figure below:

The "input" gust $w_g(t)$ induces "output" gust loads $\bar{y}(t) = \{y_1(t), y_2(t) \ldots y_n(t)\}$.

Here, the symbol $y_i(t)$, may stand for a variety of load quantities such as bending moment in a certain wing station, total tail load, c.g. vertical acceleration, etc.

We will consider aircraft with linear response properties.
The response of the output y_i is fully defined by the impulse response $h_{y_i w}(t)$:

$$y_i(t) = \int_0^t h_{y_i w}(t-\tau) w_g(\tau) d\tau .$$

$h_{y_i w}(t)$ describes the value of y_i at time t=t due to a unit impulse gust input at time t=0*

With regard to $h_{y_i w}(t)$ we may note that:

a $h_{y_i w}(t) = 0 \quad t<0$

(the system does not respond to inputs that still have to come)

b for a stable system:

$$\lim_{t \to \infty} h_{y_i w}(t) = 0.$$

---

* This unit impulse gust input is defined by:

$$w_g(t) = \delta(t) ,$$

$$\delta(t) = 0 \qquad t \neq 0$$

$$\int_{-\infty}^{+\infty} \delta(t) = 1 .$$

Hence for a stable system the Fourier transform of $h_{y_iw}(t)$ exists

$$H_{y_iw}(j\omega) = \int_{-\infty}^{+\infty} h_{y_iw}(t) e^{-j\omega t}\, dt. \tag{2.1}$$

$H_{y_iw}(j\omega)$ is called transfer function.
We may note that

- $H_{y_iw}(j\omega)$ is a complex function
  $H_{y_iw}(j\omega) = \text{Re } H_{y_iw}(\omega) + j \text{ Im } H_{y_iw}(\omega)$.
- The real part Re $H_{y_iw}(\omega)$ is a symmetric function of ω:
  Re $H_{y_iw}(\omega)$ = Re $H_{y_iw}(-\omega)$.
- The imaginary part, Im $H_{y_iw}(\omega)$ is an antisymmetric function of ω:
  Im $H_{y_iw}(\omega)$ = -Im $H_{y_iw}(-\omega)$.

Obviously the impulse response $h_{y_iw}(t)$ is the inverse Fourier transform of $H_{y_iw}(j\omega)$:

$$h_{y_iw}(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} H_{y_iw}(j\omega) e^{+j\omega t}\, d\omega. \tag{2.2}$$

Hence, if $H_{y_iw}(j\omega)$ is known, $h_{y_iw}(t)$ and hence the response $y_i(t)$ to an arbitrary gust input $w_g(t)$ can be calculated.

If our linear and stable system is excited by a sinusoidal gust input $w_g(t) = |w_g| e^{j\omega t}$, the "steady state" output response $y_i(t)$ will also be sinusoidal with frequency ω:

$$y_i(t) = |y_i| e^{j(\omega t + \varphi_i)}. \tag{2.3}$$

It can be shown that the ratio of $y_i(t)$ and input $w_g(t)$ is equal to the transfer function $H_{y_iw}(j\omega)$:

$$|y_i| e^{j(\omega t + \varphi_i)} = H_{y_iw}(j\omega)\ |w_g| e^{j\omega t}. \tag{2.4}$$

Hence $|y_i|/|w_g| = \sqrt{[\text{Re } H_i(\omega)]^2 + [\text{Im } H_i(\omega)]^2}$

$$\varphi_i = \text{arctg } \frac{\text{Im } H_i(\omega)}{\text{Re } H_i(\omega)}. \tag{2.5}$$

The transfer functions for all output loads y_i(i=1...n) with regard to the input w_g can be calculated over a range of m frequencies, and combined in one matrix H_{yw} of size mxn. The various elements in this calculation will be reviewed in the next subchapters.

2.2 Equations of motion

The response of an aircraft to turbulence can be split into two parts, namely the "rigid body" response and the "elastic" response. The rigid body response describes the resulting movement of the aircraft and consists of three translations of the c.g. plus rotations around the three axes through the c.g. of the undeformed aircraft, while the elastic response describes the resulting deformation of the aircraft under the loads applied.

In the following we will restrict ourselves to the description of the equations of motion due to a symmetrical vertical gust field, $w_g(t)$. The response to this excitation will also be symmetrical.

The equations of motion will be presented with respect to an "aircraft fixed" axis system with its origin in the c.g. of the undeformed aircraft and an x-axis coinciding with the aircraft velocity at the time t=0. (This axis system is usually called "the stability axis system", see Fig. 1.) The y-axis lies at the nose of the mean aerodynamic chord (mac), defined at b/4 here.

The rigid body response to the symmetric excitation due to gust $w_g(t)$ consists of translational or "heave" motion with velocity $\dot{z}(t)$ in z-direction and a pitch motion with angular velocity $\dot{\theta}=q$ around the y-axis (the response motion in x-direction is ignored or rather not considered).

In principle the structure may deform under load in an infinite variety of shapes. It is customary, however, to describe the aircraft elastic deformation as a linear combination of a finite number of well-defined deformation modes. The rigid body movements of the aircraft can be described by means of rigid displacement modes.

Thus, the displacement vector $\bar{w}$ of a point of the structure at location $\bar{r}$ and time t can be expressed as

$$\bar{w}(\bar{r},t) = \sum_{i=1}^{n} \bar{\phi}_i(\bar{r})\, \xi_i(t) \qquad (2.2.1)$$

where $\bar{\phi}_i(\bar{r})$ is the displacement vector of the i'th displacement mode and ξ_i is called the i'th generalised coordinate.
Here, the indices i=1 and i=2 are reserved for the rigid body modes heave and pitch respectively; hence $\dot{\xi}_1=\dot{z}$ and $\dot{\xi}_2=\dot{\theta}=q$.

The rigid body modes $\bar{\phi}_1$ and $\bar{\phi}_2$ read:

$$\left.\begin{aligned} \bar{\phi}_1^T(\bar{r}) &= (0,0,1,0,0,0) \\ \bar{\phi}_2^T(\bar{r}) &= (z,0,-x,0,1,0) \end{aligned}\right] \text{ for } \bar{r}^T = (x,y,z). \qquad (2.2.2)$$

If, in addition, n-2 different deformation modes are considered, we say that an "n-degree of freedom" (n-DOF) system is considered.

Throughout the aircraft industry it has become common practice to take as elastic deformation modes a set of "eigenmodes", that is mode shapes associated with resonance frequencies of the undamped aircraft structure.

Taking "eigenmodes" has a number of specific advantages, such as:

- Eigenmodes and frequencies are used in flutter analysis. They must be determined anyhow, and standard programmes for their determination are available.
- The eigenmodes and frequencies and associated "generalized mass" and "generalized stiffness" can also be determined experimentally.
- Due to the orthogonality of the eigenmodes, various matrices in the equations of motion become diagonal, which eases the computational effort.

However, the use of eigenmodes is not essential; any set of arbitrary deformation modes can be used and situations exist where selected deformation modes allow a more accurate description of the actual structural deformation with as few terms as possible.

For the demonstration model developed in chapter 3, three arbitrary-chosen deformation modes will be considered.

For our "n-DOF" aircraft model, a set of n equations of motion can be derived, using the so-called Lagrangian equation:

$$\frac{d}{dt}\left[\frac{\partial T}{\partial \dot{\xi}_i}\right] - \frac{\partial T}{\partial \xi_i} + \frac{\partial U}{\partial \xi_i} = Q_i \ [i=1...n] \qquad (2.2.3)$$

where T = kinetic energy in the system.
U = elastic energy in the system.
ξ_i = i^{th} generalized coordinate.

Q_i = generalized force = work performed by the external loads in a unit virtual displacement in the direction of the i^{th} coordinate.

Parts of this derivation are presented in appendix A.

The resulting set of n equations has the following form:

$$M\ddot{\bar{\xi}} + D\dot{\bar{\xi}} + K\bar{\xi} = Q_r\bar{\xi} + \bar{Q}_w w_g \tag{2.2.4}$$

with
- M = generalized mass matrix.
- D = generalized damping matrix.
- K = generalized stiffness matrix.
- Q_r and $\bar{Q}_w$ = generalized aerodynamic force matrix and vector, associated with the aircraft response motion ξ and the input gust w_g respectively.

As explained before, we are interested in the steady state response to a sinusoidal gust input $w_g=|w_g|e^{j\omega t}$; the derivatives $\ddot{\bar{\xi}}$ and $\dot{\bar{\xi}}$ can then be expressed as $-\omega^2\bar{\xi}$ and $j\omega\bar{\xi}$ respectively, and (2.2.4) can be rewritten

$$[-\omega^2 M + j\omega D + K - Q_r]\,\bar{\xi} = \bar{Q}_w w_g\ . \tag{2.2.5}$$

Note that generally the elements of $\bar{Q}_w$ and Q_r are complex functions of the gust frequency ω.

Equation (2.2.5) can be written in simplified form as

$$A\bar{\xi} = \bar{Q}_w w_g$$

with $A = -\omega^2 M + j\omega D + K - Q_r$.

The solution of the equations of motion reads:

$$\bar{\xi} = A^{-1}\ .\ \bar{Q}_w w_g. \tag{2.2.6}$$

2.3 The loads equation

As explained before, one may define an arbitrary set of output "Loads" y_i, for example beding, shear, torsion in various wing sections, c.g. acceleration, tail load etc.
We will assume that each load y_i can be expressed as a linear function of the input load w_g and the generalized coordinates $\bar{\xi}$, or, in matrix notation:

$$\bar{y} = C\bar{\xi} + \bar{D}w_g. \tag{2.3.1}$$

Combining (2.2.11) and (2.3.1) yields:

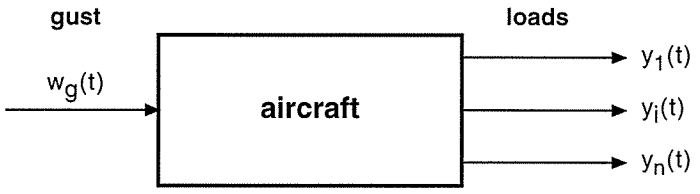
$$\bar{y} = [C.A^{-1}.\bar{Q}_w + \bar{D}]\, w_g \text{ , or}$$
$$\bar{y} = \bar{H}\, w_g \tag{2.3.2a}$$

where the transfer function vector $\bar{H}$ for one frequency is equal to

$$\bar{H} = C\, A^{-1}\, \bar{Q}_w + \bar{D} \quad . \tag{2.3.2b}$$

2.4 Basic PSD-equations

Consider again the response scheme of section 2.1:



Suppose the input gust $w_g(t)$ is a stationary random process $\underline{w}_g(t)$ having a Gaussian probability distribution with standard deviation σ_w and normalized power spectral density function $\Phi_w^n(\omega)$.

Then each output $y_i(t)$ is also random, with Gaussian probability distribution and standard deviation σ_{y_i}.

σ_{y_i} may be calculated from:

$$\sigma_{y_i}^2 = \sigma_w^2 \int_{-\infty}^{\infty} |H_{y_i w}(j\omega)|^2 \, \Phi_w^n(\omega) \, d\omega \tag{2.4.1}$$

$$\text{or} \quad \sigma_{y_i} = \bar{A}_{y_i} \, \sigma_w \tag{2.4.2}$$

$$\text{with } \bar{A}_{y_i} = \left[\int_{-\infty}^{\infty} |H_{y_i w}(j\omega)|^2 \, \Phi_w^n(\omega) \, d\omega \right]^{1/2}. \tag{2.4.3}$$

The correlation function between output $y_i(t)$ and $y_j(t)$ is defined as:

$$\rho_{y_i y_j} = \frac{\int_{-\infty}^{\infty} \left[H_{y_i w}(j\omega) \, H_{y_j w}^*(j\omega) \right] \Phi_w^n(\omega) \, d\omega}{\sigma_{y_i} \, \sigma_{y_j}} = \frac{\int_{-\infty}^{\infty} \mathrm{Re} \left[H_{y_i w}(j\omega) \, H_{y_j w}^*(j\omega) \right] \Phi_w^n(\omega) \, d\omega}{\sigma_{y_i} \, \sigma_{y_j}}. \tag{2.4.4}$$

Note: $-1 \le \rho_{y_i y_j} \le 1$;
if $|\rho_{y_i y_j}| \approx 1$, the loads y_i and y_j are said to be highly correlated; if $\rho_{y_i y_j} = 0$, these loads are uncorrelated.

Another parameter used in PSD calculations is the so-called N(0) value or "number of positive zero crossings". N(0) for y_i is calculated from

$$N(0)_{y_i} = \frac{1}{2\pi} \frac{\left[\int_{-\infty}^{\infty} \omega^2 |H_{y_i w}(j\omega)|^2 \, \Phi_w^n(\omega) \, d\omega \right]^{1/2}}{\bar{A}_{y_i}}. \tag{2.4.5}$$

3 Program development

3.1 Functional Requirements

The program is intended for educational purposes as well as for professional purposes; in order to be available for as many users as possible the program has been prepared for use in a PC-environment with the Student edition of MATLAB4 for Windows. This imposes a restriction on the size of the largest matrices that can be handled and hence on the complexity of the aircraft model that can be treated.

Also, to avoid boredom and loss of user's interest, the various calculation-options offered in the program menu should be executable in a reasonably short time, say less than one minute on a PC with 150 MHz Pentium processor.

In order to provide a clear insight in the aircraft response model, the computational representation is kept simple. Solution procedures in the frequency domain in combination with Fourier transforms will be used rather than direct integration in the time domain. Because of this, the program is only applicable for studying the response of linear systems.

As said before, the program is intended to study various aspects of aircraft response to turbulence. The effects of various parameters can be evaluated by carrying out successive response calculations, in which the relevant parameters are varied. The program should offer the possibility to study the following aspects:

a. The Aircraft response properties.
   - Rigid body response behaviour.
     The program must allow the evaluation of the effect of changes in rigid body response properties on gust induced loads. This includes comparison of heave/pitch- and heave only- response freedom.
     Parameters to be variable in program are:
     - A.C. size and geometry (wing/tail area, tail arm, sweepback etc).
     - A.C. mass and Inertia, and c.g. position.
     - Basic aerodynamics (C_{Z_α}, inclusion aerodynamic inertia).
   - Elastic response behaviour.
     The program should allow evaluation of the effect of deformations of various structural components on gust induced loads. This includes the effect on the aerodynamic load distribution (static aeroelastic effect) as well as the inertia-loads (dynamic effect).
     Specifically, wing bending and wing torsion (large effect on load distribution as well as inertia loads) and rear fuselage bending (reduction of tail-efficiency) should be included.
     It should be possible to suppress the respective deformation modes separately.

- Output loads.
  The program should calculate output loads for a number of relevant load quantities. The following output loads are considered of primary interest:
  - Centre of gravity acceleration (in g).
  - Shear force (positive downward), bending moment (positive tip down), and torsion moment (positive leading edge up) in wing root.
  - Total tail load (positive downward).

b. Input loading cases.
The program must be capable to calculate loads for the loading cases specified in the current civil airworthiness requirements.

i. Discrete Gust:
(1-cos) discrete gust shape (in accordance with FAR/JAR25).
- Variables are: gust length and gust strength.

Output: Loads as function of time.

ii. Continuous Gust:

- PSD gust calculation:
  Input: "von Karman" gust PSD function.
  Output: for each output: PSD-function, $\bar{A}$, N(0), and relevant correlation functions.
- "real time" continuous gust: Although currently not included in airworthiness requirements, it is informative to visualize in the time domain the response to continuous random turbulence.
  The program must provide the possibility to calculate response to a random gust signal with "von Karman"PSD function. Output: Load-time traces for all output loads.

3.2 Aircraft model description

A simple aircraft model with 5 degrees of freedom has been defined.

In the following, a full description of this model will be presented.

3.2.1 Aircraft planform

- The aircraft model is two-dimensional or "flat", that means no dimensions in z-direction are considered.
- The aircraft "structure" consists of:
  - A beam-model fuselage.
  - A prismatic wing with sweepback angle lambda.
  - A straight prismatic stabilizer.

The aircraft planform is shown in figure 1.
The "default" values for the aircraft dimensions and main mass properties are given in table 3.1

Basic aerodynamics

Simple strip theory aerodynamics are applied with constant C_{z_α} along the wing span (czaw) and along the tail span (czat). The aerodynamic loading on the fuselage is taken into account by applying a fuselage aerodynamic moment, characterized by a constant moment coefficient cmaf. The downwash angle ε at the tail is characterized by downwash coefficient $d\varepsilon/d\alpha$.
The "default" values are given in table 3.2

Aerodynamic inertia is taken into account by applying Theodorsen's function to aircraft response induced incidence angle, and Sears function to gust induced incidence angle.

The expressions used are (Ref. 2):

$$\phi_{theo}(j\omega) = \frac{\left(-0.5\ \omega^2 + .56085\ j\omega\ \frac{V}{\bar{c}} + .054\ \frac{V^2}{\bar{c}^2}\right)}{\left(.09\ \frac{V}{\bar{c}} + j\omega\right).\left(.6\ \frac{V}{\bar{c}} + j\omega\right)} \qquad (3.2.1)$$

$$\phi_{sear}(j\omega) = \frac{\left(1.13\ j\omega\ .\ \frac{V}{\bar{c}} + .52\ \frac{V^2}{\bar{c}^2}\right)}{\left(.26\ \frac{V}{\bar{c}} + j\omega\right)\left(2\ \frac{V}{\bar{c}} + j\omega\right)} \qquad (3.2.2)$$

with V = aircraft speed (TAS).
$\bar{c}$ = mean aerodynamic chord.

Note that Mach-effects are not accounted for in these expressions.
The calculation of the airload on a wingstrip is described in some more detail under paragraph 3.2.6.

3.2.2 Degrees of freedom

Five degrees of freedom are considered , two associated with the "rigid body" motions heave and pitch, and three with elastic deformation modes.

The elastic deformation modes are "partial" in this sense that each of them only describes the deformation of a part of the structure. The deformation modes are given as simple mathematical expressions. (Note that the deformation modes are not eigenmodes; they are not orthogonal with respect to the rigid body modes or with respect to each other.)

In the present model, the deformation modes are "predefined"; they cannot be changed easily. The elastic degrees of freedom and associated modeshapes are (see also Fig. 2):

- Mode 3: Rear fuselage bending.
  The modeshape is given by the following simple polynomial of the 3rd order:
  (see Fig. 2a)

$$w_3 = \frac{1}{2}(xl)^2(3-xl) \qquad (3.2.3)$$

  where the "local coordinate" xl runs from 0 at the aircraft axis origin to 1 at the location of the horizontal tail.
  Note that w_3 is zero at the axis origin (x=0), and 1 at the tail.
  The second derivative of w is a linear function of x, with w"=0 at the tail;hence, the modeshape corresponds with the deflection of a prismatic bar clamped at x=0 and loaded by a point load at the tail.
- Mode 4: Wing bending.
  The mode shape is given by a 4th order polynomial (see Fig. 2b)

$$w_4 = \frac{1}{3}((1-xl)^4-4(1-xl)+3) \qquad (3.2.4)$$

  where the "local coordinate"xl runs from 0 at the wing root to 1 at the wing tip.
  The modeshape corresponds with the deflection of a prismatic bar, clamped at the wing root and loaded by an evenly distributed load. As the wing has a bending stiffness EI that increases toward the wing root, the chosen modeshape corresponds with the wing deflection due to a distributed load that increases from tip to root.
- Mode 5: Wing torsion.
  The mode shape is given by the simple parabolic expression (see Fig. 2c):

$$\psi_5 = 2xl-xl^2 \qquad (3.2.5)$$

  where again the "local coordinate" xl runs from 0 at the root to 1 at the wing tip. As the torsional stiffness GJ of the wing increases towards the wing root, this torsional mode roughly corresponds with the deflection of the wing under a distributed torsional loading that increases towards the root.

3.2.3 Calculation of generalized mass and damping matrix

To calculate the generalized masses, the rear fuselage and each wing half have been split up in 10 and 5 elements respectively, see figure 3. A wing element has mass m_l, and moment of inertia about the local coordinate system I_{xl} and I_{yl}. A fuselage element has the mass m_l and inertia I_{yl}. The horizontal tail is modelled by one element having inertia properties m_l and I_{xl}.

The aircraft equations of motion (eq. 2.2.4) describe the response with respect to an axis system that is fixed to the aircraft. Apart from mass coupling elements in the generalized mass matrix, also damping contributions arise in the generalized damping matrix, due to the non-orthogonality of the displacement modes chosen here in subchapter 3.2.2. To clarify this, the derivation of matrices M and D from the Lagrangian equations is discussed in more detail in appendix A.

Assuming the aircraft to consist of k elements, a generalized mass matrix (size nxn for n degrees of freedom) element M_{ij} is computed from:

$$M_{ij} = \sum_{\text{all } k} \bar{\phi}_{i,k}^{T} [m]_k \bar{\phi}_{j,k} \tag{3.2.6}$$

where $[m]_k$ = mass matrix of element k
$\bar{\phi}_{i,k}$ = mode shape vector in element k.

The element mass matrix is a matrix with the mass and inertia properties of element k. The vector $\bar{\phi}_{i,k}$ contains the relevant displacement and rotiations in the centre of the element k, determined from mode shape vector $\bar{\phi}_i$.

The size of the damping matrix D is nxn; this matrix is associated with the rotational movement of the considered aircraft-fixed axis system.

The element D_{ij} is calculated from:

$$\left.\begin{aligned} D_{ij} &= 0 \qquad j \neq 2 \\ D_{i2} &= -V \sum_{\text{all } k} \bar{\phi}_{i,k}^{T} [m]_k \bar{\phi}_{1,k} \end{aligned}\right\}. \tag{3.2.7}$$

3.2.4 Calculation of the generalized stiffness matrix

For the calculation of the generalized stiffnesses the aircraft is divided in the same elements as used for the determination of the generalized masses (see Fig. 3). A wing element has a constant bending stiffness EI and a constant torsional stiffness GJ. A fuselage element only has a constant bending stiffness EI. The tail element is rigid.

For an element of length Δx, the elastic energy contained in the beam element is given by

$$U = \frac{1}{2}\int_0^{\Delta x} \frac{M_y^2(x)}{(EI)_x}\,dx + \frac{1}{2}\int_0^{\Delta x} \frac{M_x^2(x)}{(GJ)_x}\,dx,$$

or, as $(EI)_x\, w''(x) = M_y(x)$ and

$$(GJ)_x\, \psi'(x) = M_x(x), \qquad (3.2.8)$$

$$U = \frac{1}{2}\int_0^{\Delta x} EI[w''(x)]^2\,dx + \frac{1}{2}\int_0^{\Delta x} GJ[\psi'(x)]^2\,dx.$$

where w = displacement in z-direction (w'' is second derivative to x).
ψ = torsional rotation (ψ' is first derivative to x).

The beam elements are modelled such, that the bending moment M_y varies linearly along the length Δx, while the torsion moment M_x is a constant along Δx. This means that w'' varies linearly and ψ' is constant along the element. The above integrals can then be solved to yield the following simple expression, which is a function of w"(l), w"(r), and ψ', where the indices l and r denote the left-hand extremity and the right-hand extremity of the element respectively. The value of ψ' is taken at the element mid-point.

$$U = \frac{EI}{6}\Delta x((w''(l))^2 + w''(l)w''(r) + (w''(r))^2) + \frac{GJ}{2}\Delta x(\psi')^2. \qquad (3.2.9)$$

Applying the equation of Lagrange, we can derive the generalized stiffnesses. For element k the values (l and r) of the second derivative of the bending displacement by mode shape i are stored in the vector $\bar{\phi}''_{i,k}$, and the first derivative of the torsional rotation by mode shape i is stored in the one-element vector $\bar{\phi}'_{i,k}$. A generalized stiffness matrix element is calculated with:

$$K_{ij} = \sum_{\text{all } k} \left[\bar{\phi}^{//\,T}_{i,k} [k_b]_k \bar{\phi}^{//}_{j,k} + \bar{\phi}^{/\,T}_{i,k} [k_t]_k \bar{\phi}^{/}_{j,k} \right]. \qquad (3.2.10)$$

where $[k_b]_k$ = bending stiffness matrix of element k:

$$[k_b]_k = \frac{EI\,\Delta x}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$[k_t]_k$ = torsional stiffness matrix of element k: $GJ\Delta x$ (contains only one element).

It is usual to include so-called structural damping (sd) by adding an imaginary component to K_{ij}:

$$K^*_{ij} = K_{ij}[1 + j.sd].$$

A typical value for sd = 0.03.

It can easily be shown that for the 5 displacement modes considered in our model,

$$K_{ij} = 0, \; i \neq j.$$

Also, $K_{11} = K_{22} = 0$, as rigid modes don't result in elastic energy.
Hence, the generalized stiffness matrix contains only three diagonal terms K_{33}, K_{44} and K_{55}.

3.2.5 Calculation of generalized aerodynamic matrices

The elements of the nxn-matrix Q_r in equation 2.2.4 are calculated from:

$$Q_r(ij) = \sum_{\text{all } k} \bar{\phi}^{T}_{i,k} \bar{F}_{j,k} \qquad (3.2.11)$$

where $\bar{F}_{j,k}$ represents the aerodynamic force and moment on element k associated with a unit displacement of the mode j, and $\bar{\phi}_{i,k}$ contains the relevant displacement and rotation due to mode i.

As we consider only one input w_g, the matrix Q_w has size (nx1), the element $Q_w(i)$ is calculated from:

$$Q_w(i) = \sum_{\text{all } k} \bar{Q}_{i,k}^T \bar{F}_{w,k}. \qquad (3.2.12)$$

Note that generally the elements of Q_w and Q_r are complex functions of the gust frequency ω.

The calculation of the aerodynamic forces and moments in $\bar{F}$ is treated in the next subchapter.

3.2.6 Calculation of airloads on wing and tail

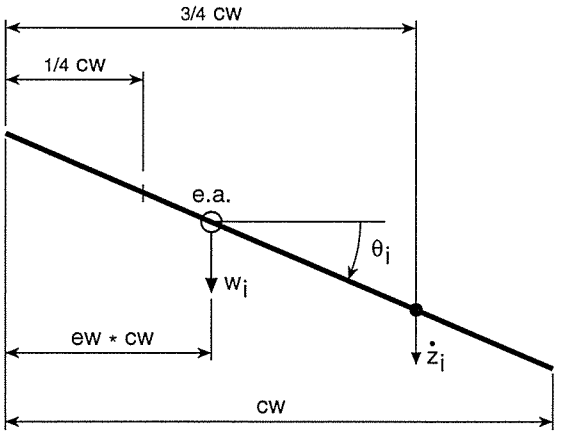
Each wing half is split up in five strips with width $\frac{b}{10}$; the horizontal tail is considered as one strip with width bt, see figure 4.

The airload on a wing strip i is calculated from:

$$\Delta L_i = \frac{b}{10} \cdot cw \cdot \frac{1}{2}\rho V^2 \left[\alpha_{w_i} \phi_{theo} + \alpha_{g_i} \phi_{sear}\right] C_{z_\alpha}.$$

Here, α_{g_i} is the gust induced angle of attack, and α_{w_i} is the angle of attack of wing strip i due to aircraft response.

The airload ΔL_i acts in the 1/4-chord point; the angle α_{w_i} is defined by the downward velocity of the 3/4-chord point $\dot{z}_i$ and by the pitch angle of the strip due to elastic deformation $\theta_{i_{elastic}}$:



$$\alpha_{w_i} = \frac{\dot{z}_i}{V} + \theta_{i_{elastic}} = \frac{\dot{w}_i + cw(\frac{3}{4} - ew)\,\dot{\theta}_i}{V} + \theta_{i_{elastic}}. \tag{3.2.13}$$

ΔL_i induces a moment ΔM_i with respect to the elastic axis:

$$\Delta M_i = (ew - 1/4)\ cw * \Delta L_i. \tag{3.2.14}$$

In addition, an aerodynamic moment $\Delta M_{\dot{\theta}_i}$ occurs, which is specifically related to $\dot{\theta}_i$ ("apparent profile camber") (Ref. 3).

$$\Delta M_{\dot{\theta}_i} = \frac{-b}{10} \cdot cw \cdot \frac{1}{2} \rho V^2 C_{z_\alpha} \cdot \frac{1}{16} \frac{(cw)^2}{V} \cdot \dot{\theta}_i \cdot \phi_{theo}. \tag{3.2.15}$$

The calculation of the airload on the tail is comparable, except for the fact that the angle of attack $\alpha_t(t)$ must be decreased by the downwash angle $\alpha_{t_{down}}(t)$, which is equal to the downwash coefficient times the angle of attack of the second wing element α_{w_2} at the time $t' = t - l_t/V$:

$$\alpha_{t_{down}}(t) = \frac{d\varepsilon}{d\alpha} \left(\alpha_{w_2} + \alpha_{g_2}\right)_{t'=t-l_t/V}. \tag{3.2.16}$$

where l_t = distance between second wing strip and tail.

In the frequency domain, this time delay is represented by an exponential delay function:

$$\alpha_{t_{down}}(j\omega) = \frac{d\varepsilon}{d\alpha} \left(\alpha_{w_2}(j\omega) + \alpha_{g_2}(j\omega)\right) e^{-j\omega\frac{l_t}{V}}. \tag{3.2.17}$$

With regard to the gust angle of attack $\alpha_g(t)$, it must be noted that the model assumes a simple one dimensional "wash board" type gust field. Hence, in case of sweep back the various wing strips are hit by the gust at different times.

Taking the time when the most inboard wing strip (the first wing strip) is hit as reference, the following relations hold:
for wing strip i:

$$\left.\begin{aligned} &\alpha_{g_i}(t) = \alpha_g(t-(i-1)\,\tau) \quad (i=1...5) \\ &\text{with } \tau = \frac{b.\tan(\Lambda)}{10\ V} \\ &\text{for the tail:} \\ &\alpha_{g_t} = \alpha_g(t-\tau_t) \\ &\text{with } \tau_t = \left[xt + \frac{b.\tan(\Lambda)}{5}\right]/V \end{aligned}\right\} \qquad (3.2.18)$$

In the frequency domain:

$$\left.\begin{aligned} &\alpha_{g_i}(j\omega) = \alpha_g(j\omega)\ e^{-(i-1)j\omega\tau} \quad (i=1...5) \\ &\alpha_{g_t}(j\omega) = \alpha_g(j\omega)\ e^{-j\omega\tau_t} \end{aligned}\right\} \qquad (3.2.19)$$

3.3 Gust input description

The program offers the possibility to carry out the following calculations:

a Load-time response to a single gust with (1-cos) shape.

b Calculation of the $\bar{A}$-values and N(0)-values for all output loads plus correlations $\rho_{y_i y_j}$ for a number of selected outputs.

c Load time response to a patch of randomly varying turbulence.

ad a:
The input gust is given in the time domain as:

$$w_g(t) = \frac{w_{g_{max}}}{2}\left[1 - \cos\left(2\pi\,\frac{t}{t_g}\right)\right] \qquad 0 < t < t_g,\ \text{else } w_g(t) = 0 \qquad (3.3.1)$$

$$\text{with } t_g = \frac{l_g c w}{V}$$

here $w_{g_{max}}$ is the gust strength and l_g is the gust length, expressed in wing chords.

The Fourier transform of $w_g(t)$ reads:

$$W_g(j\omega) = \frac{w_{g_{max}}}{2}\left[\frac{1-e^{-j\omega t_g}}{j\omega} + \frac{j\omega(e^{-j\omega t_g}-1)}{\left(\frac{2\pi}{t_g}\right)^2 - \omega^2}\right]. \qquad (3.3.2)$$

The time response for output y_i is found by an inverse Fourier transform:

$$y_i(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} W_g(j\omega)\, H_{y_i w}(j\omega)\, e^{j\omega t} d\omega. \qquad (3.3.3)$$

ad b:
The shape of the turbulence power spectrum is the usual "von Karman"-expression

$$\Phi_w^n(\omega) = \frac{L\left(1+\frac{8}{3}(1.339\,\omega\frac{L}{V})^2\right)}{2\pi V\left(1+\left(1.339\,\omega\frac{L}{V}\right)^2\right)^{11/6}} \qquad (3.3.4)$$

L = scale of turbulence = 762 m (2500 ft).

ad c:
A stochastic gust signal in the frequency domain is generated

$$\underline{W}_g(j\omega) = \sigma_w \sqrt{\Phi_w^n(j\omega)}\; e^{j\varphi(\omega)}$$

where σ_w is the intensity of the turbulence patch and $\varphi(\omega)$ is random between 0 and 2π.

The time response $\underline{y}_i$ to this signal is a stochastic process calculated from

$$\underline{y}_i(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \underline{W}_g(j\omega)\, H_{y_i w}(j\omega)\, e^{j\omega t} d\omega.$$

4 How to use the program

An insight in the program structure can be gained from figure 5. This diagram shows in what order the different subroutines are called in the program "GRM", the Gust Response Model. A complete program listing is given in appendix B.

The program starts by typing GRM at the Matlab command line. The user is then prompted to make some basic choices with regard to the model to be created: the number of degrees of freedom, and the representation (yes or no) of aerodynamic inertia (Theodorsen and Sears functions).

On the basis of the above basic choices, a default aircraft gust response model will be generated by the routine CREAMOD. During this generation process, default aircraft characteristics such as planform, aircraft mass, and lift coefficients are displayed (in the routine ACDATA) and can be changed by the user. To influence the flexible modes of the model, the user can also multiply mass and inertia distributions along wing and rear fuselage by a factor; the elements in the stiffness matrix for the flexible modes can also be multiplied by a user-defined factor.
The default aircraft characteristics are comparable to a medium-sized jet transport aircraft.

Hints

If you want to remove one elastic mode from the model, this can be done by applying a large multiplication factor (e.g. 1000) to the corresponding element in the (diagonal) stiffness matrix. The model can now be considered "infinitely stiff" in this displacement mode. The numbering of the displacement modes can be found in subchapter 3.2.2.

To investigate the influence of aerodynamic forces due to elastic deformations, the inertia forces due to elastic motions can be deleted. This can be done by applying a very small multiplication factor (e.g. 0.001) to the mass/inertia distribution of the concerning aircraft part. Dynamic inertia forces will then be negligible with respect to the aerodynamic forces due to the considered mode.

Using the aircraft data from ACDATA and the degrees of freedom with pre-defined displacement functions (see 3.2.2), the generalized mass, damping, stiffness, and aerodynamic matrices are generated. Subsequently, the value of the transfer function of each output is calculated according to equation (2.3.2b) for a range of frequency values. The output load quantities of the model are:

- Load factor Δn_z.
- Shear force, bending moment, and torsion moment in wing root.
- Shear force in horizontal tail root.

Note that wing and tail roots in this case are on the aircraft centerline.

The program now returns with a menu, from which the user can choose from the following actions:

1. Calculate and plot time response to (1-cos) gust.
2. Change parameters and calculate new transfer functions.
3. Calculate $\bar{A}$, ρ, and plot power spectra of all outputs.
4. Calculate response to stochastic gust patch.
5. Plot transfer functions.
6. Quit.
7. Give keyboard control.

ad 1. The user defines maximum gust speed (m/s TAS) and the total length (in number of chord lengths) of the gust bump (routine TRESP). Responses are plotted on the screen (routine PLTRESPS). The results of a possible previous calculation are represented by a green dashed line, the present results by a yellow line. See the example in figures 6a-6b.

ad 2. The basic model characteristics such as degrees of freedom and planform can be changed (the program returns to CREAMOD).

ad 3. The routine ABARGRM calculates and displays the $\bar{A}$ values of all output quantities, and the correlation coefficients between load factor and wing bending, and between wing shear and wing torsion. The power spectra (PSD's) of the outputs (of the present and a possible previous calculation) are plotted on the screen (PLTSPECS). Note that the load spectra are multiplied by the frequency, so that the area under the logarithmic graph is the total power of the output: $\int_0^\infty \Phi_{yy}(f)\,df = \int_0^\infty f\Phi_{yy}(f)\; d(\log(f))$.
See the example in figures 7a-7b.

ad 4. The response to a stochastic gust patch (STCHRESP) takes quite some calculation time, due to the fact that the transfer functions are generated once again, at more frequency values (and equidistant) than in the first part of the program. This equidistant frequency distribution is necessary in order to get a good Gaussian distribution of the stochastic gust signal. Aircraft responses are calculated to an infinite, periodic patch of stochastic turbulence with period 34 s, and the first 10 seconds of the period are plotted on the screen (PLTSTOCH). See the example in figures 8a-8b.

ad 5. The transfer functions of the present and the possible previous aircraft model are plotted on the screen by PLTTFFS. See the example in figure 9.

ad 6. Exit the program, clear all variables.

ad 7. This option allows to view and/or change parameters when an aircraft model has been created; parts of the program can also be rerun (for advanced users).

Hints

As many more interesting effects can be thought of to investigate than the options allowed during running the program, the advanced user is invited to simply modify the m-files to her/his purposes. For instance, if we want to visualise the contribution of an elastic mode's inertia forces to the total load outputs we first calculate the default model transfer functions. After this, a new model is generated, this time applying a multiplication factor 0.001 to the corresponding mass/inertia distributions and a factor 1000 to the corresponding column of the generalized mass matrix. In this way, the inertia forces are removed only from the output equations (GENOUT) of the model; the modal responses of the aircraft remain unchanged. Applying a multiplication factor to a column in the mass matrix is not a standard option in the program, so the user has to adapt the GENMASS m-file for this calculation. If we now run PLTTFFS, the difference between the yellow line (present results) and the green dashed line (previous results) is caused by inertia forces from the considered mode in the previous results.

The open structure of m-files makes it possible to customize the program code (don't forget to make a back-up).

As discussed in subchapter 3.1, the GRM program represents the aircraft gust response characteristics by means of frequency response functions. A limited frequency range has been implemented as a default, and it has to be kept in mind that signals with high-frequency components cannot be represented well in the time domain due to the limited Fourier transform. For instance a (1-cos) gust signal shows irregularities in the time domain representation if the gust bump is shorter than about 8 chords in the present default conditions.

Another disadvantage of the frequency domain concept is that instability cannot be detected very well. An indication of instability is however when the time response of an output does not start at zero for t=0.

5 Concluding remarks

1 A "MATLAB" computer program has been developed to study various aspects of aircraft response to vertical turbulence.

2 The program includes a simple aircraft model with 5 symmetrical degrees of freedom. The various parameters in this model can be varied.

3 The types of gust input that can be simulated include (1-cos) discrete gusts as well as patches of continuous turbulence. Apart from that, "standard" PSD calculations can be performed.

4 The program is limited to the study of aircraft with linear response characteristics.

5 This report is intended as part of a "Manual" on aircraft loads.

6 References

1. Jonge, J.B. de; *Tail Loads in Pitching Manoeuvres*, NLR CR 96081 L.
2. Bisplinghoff, R.L.; Ashley, H.; Halfman, R.L.; *Aeroelasticity*, Addison-Wesley Publishing Company, Inc., Massachusetts, 1957.
3. Fung, Y.C.; *An Introduction to the Theory of Aeroelasticity*, John Wiley & Sons, Inc. New York, 1955.

Table 3.1 Aircraft dimensions and main mass properties

property	name in program	default value	description
cw	cw	3.83 m	wing chord (= mean aerodynamic chord)
b	b	24 m	wing span
Λ	labda	17°	sweepback angle
ew	ew	0.35	wing elastic axis position in wing chord lengths
ct	ct	2.29 m	stabilizer chord
bt	bt	10 m	stabilizer span
xt	xt	17 m	distance between stabilizer e.a. and Origin in nose of mac
et	et	0.25	tail elastic axis position in tail chord lengths
$c_{x,cg}$	cxcg	0.15	centre of gravity position behind nose of m.a.c, in wing chords
x_{cg}	xcg	-cw*cxcg	center of gravity position in meters (see fig 1; xcg is negative when cg lies behind nose of m.a.c.)
S_w	Sw	cw*b	wing surface area
S_t	St	ct*bt	horizontal tail surface area
m	m	20e3 kg	half aircraft mass
I_y	Iy	8.122e5 kgm^2	half aircraft moment of inertia around lateral axis

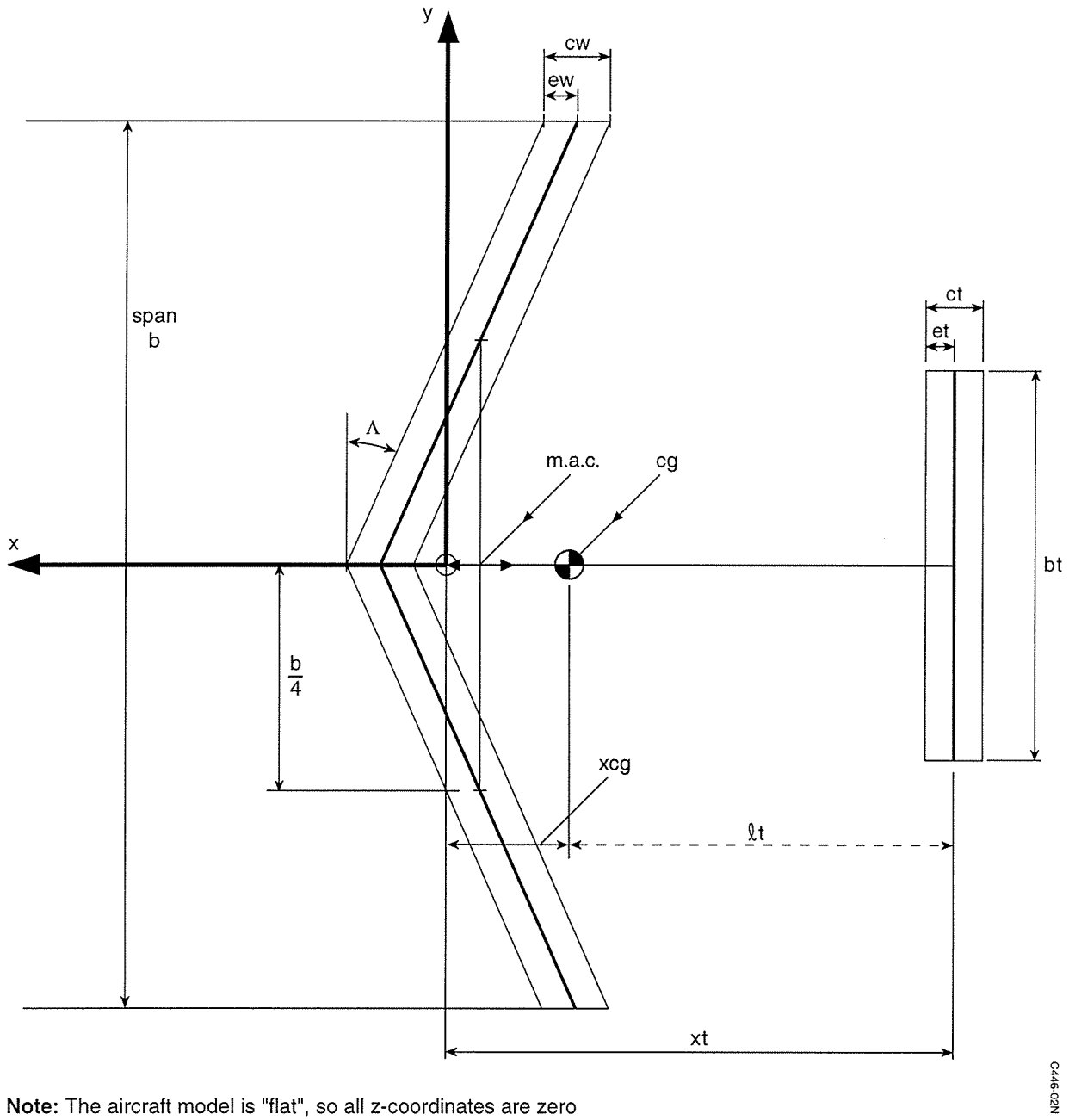
Note that the numerical values given are "default" values, that can be changed independently.

Table 3.2 Aircraft aerodynamic properties

property	name in program	default value	description
C_{z_α}	czaw	-6.379	wing strips lift coefficient (for Λ=0)
	cZaw	czaw*cos(labda)	wing strips lift coefficient in airflow direction
$C_{z_{\alpha_t}}$	czat	-4.61	stabilizer strips lift coefficient
$d\varepsilon/d\alpha$	deda	0.35	downwash coefficient
$C_{m_{\alpha_f}}$	cmaf	0.4	fuselage moment coefficient
V	V	220 m/s	aircraft speed (TAS)
ρ	rho	0.59 kg/m^3	air density

Table 3.3 Inertia and stiffness of the fuselage- wing-and tail-elements

Component	Element nr. (Fig. 3)	Inertia			Stiffness	
		m_l (*1e2 kg)	I_{xl} (*1e2 kg m^2)	I_{yl} (*1e2 kg m^2)	EI (*1e7 Nm^2)	GJ (*1e6 Nm^2)
Fuselage	f1	8.9160		14.470	132.0	
	f2	1.8750		0.262	99.5	
	f3	6.8150		5.013	85.5	
	f4	7.3110		6.634	79.5	
	f5	2.1870		3.649	73.5	
	f6	4.1890		4.216	57.5	
	f7	1.0110		1.309	47.0	
	f8	0.8874		7.010	35.0	
	f9	0.8924		7.470	24.0	
	f10	2.3000		30.000	9.5	
Wing	w1	20.0	10.333	54.458	16.900	128.00
	w2	16.0	8.267	43.566	9.520	64.80
	w3	12.0	6.200	32.675	3.450	22.80
	w4	8.0	4.133	21.783	1.210	8.10
	w5	4.0	2.067	10.892	0.490	3.30
Tail	t1	2.90	1.35			



Note: The aircraft model is "flat", so all z-coordinates are zero

Fig. 1 Planform of aircraft model

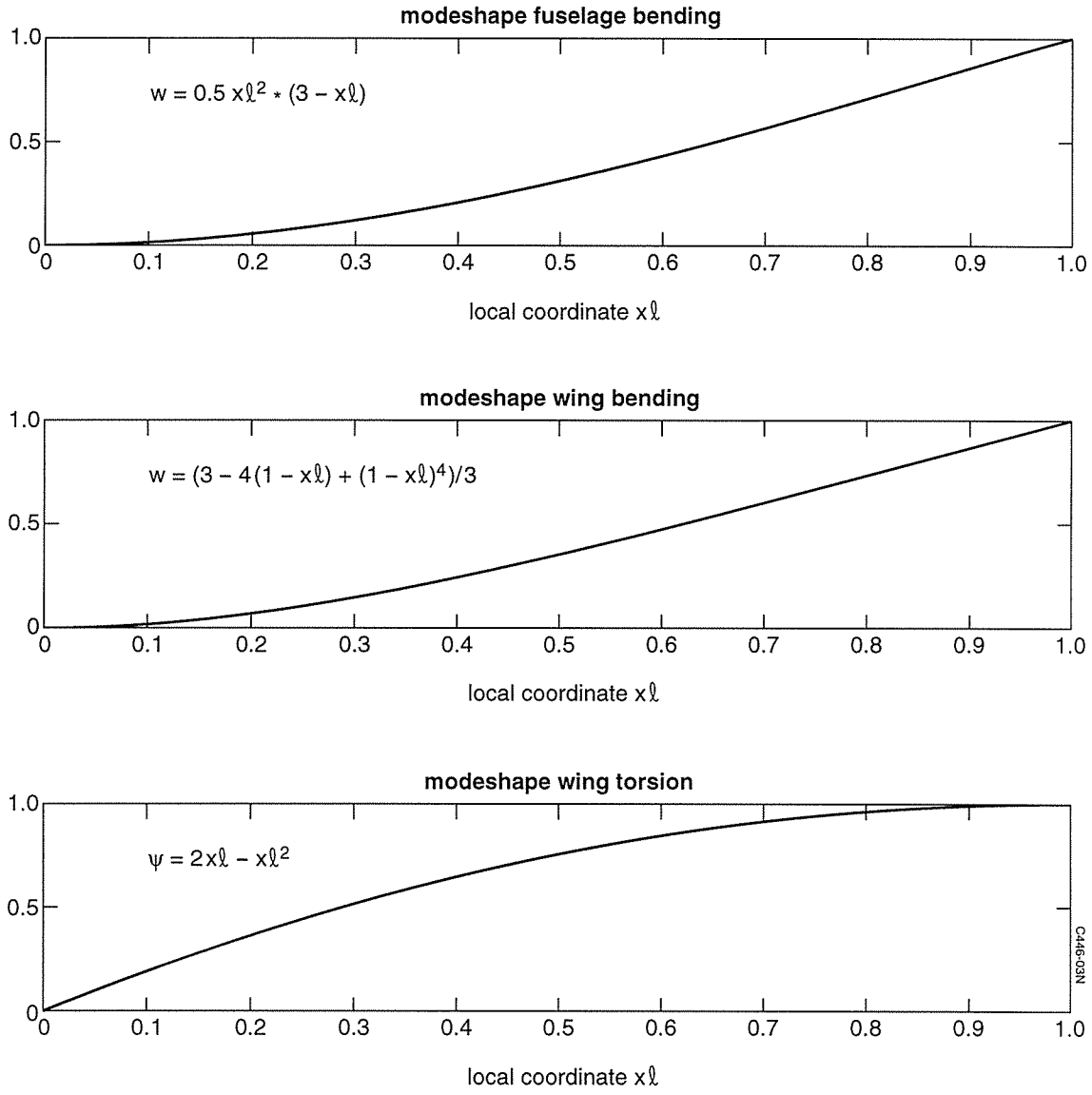


Fig. 2 Elastic structural displacement modeshapes

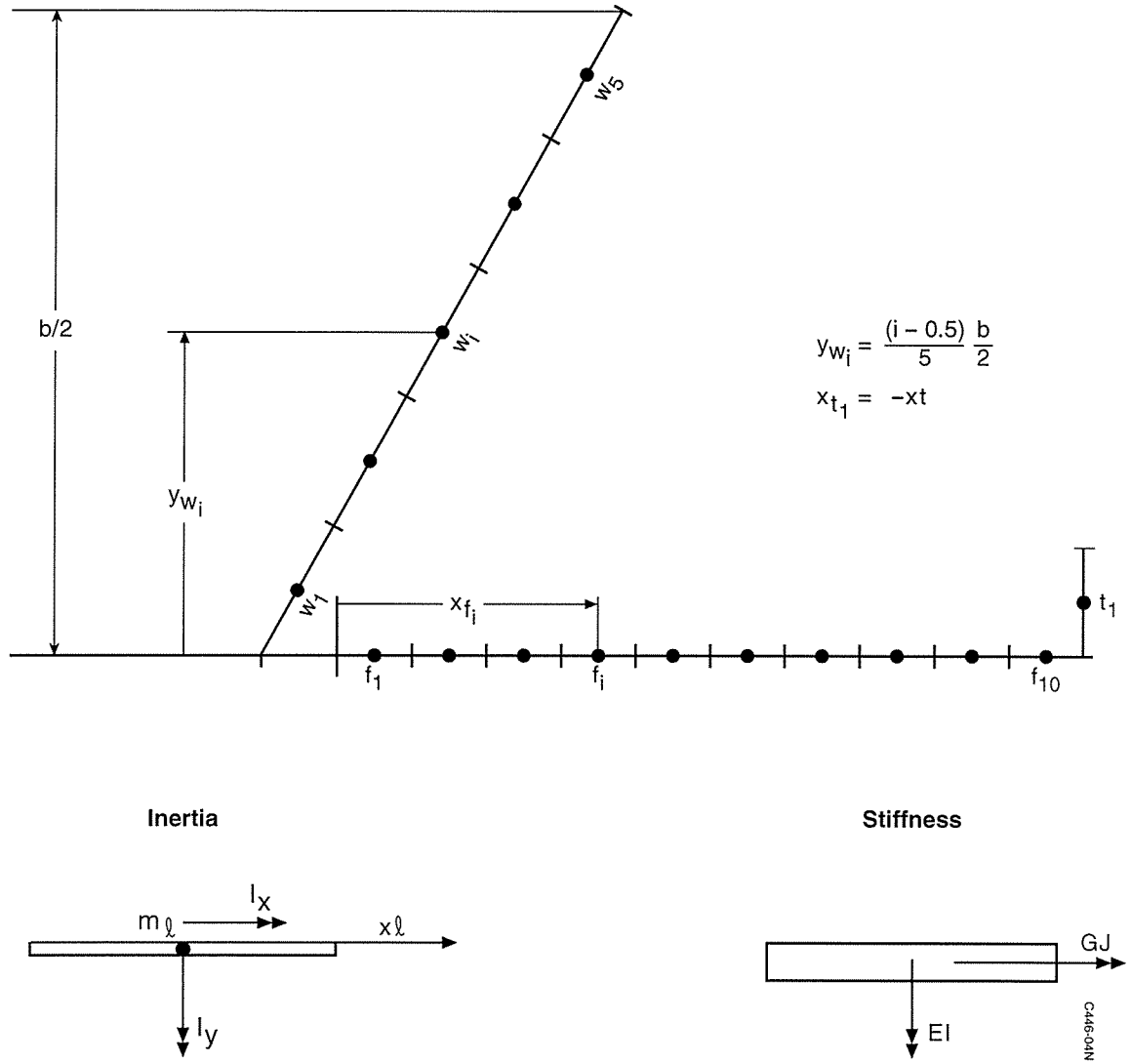


Fig. 3 Lumping of fuselage and wing

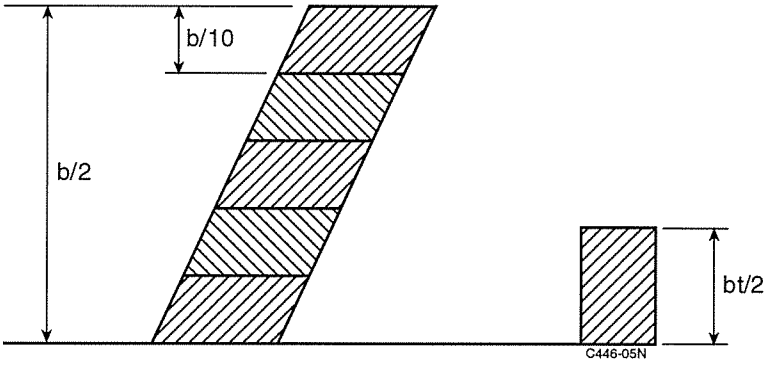


Fig. 4 Distribution of lifting surfaces in strips

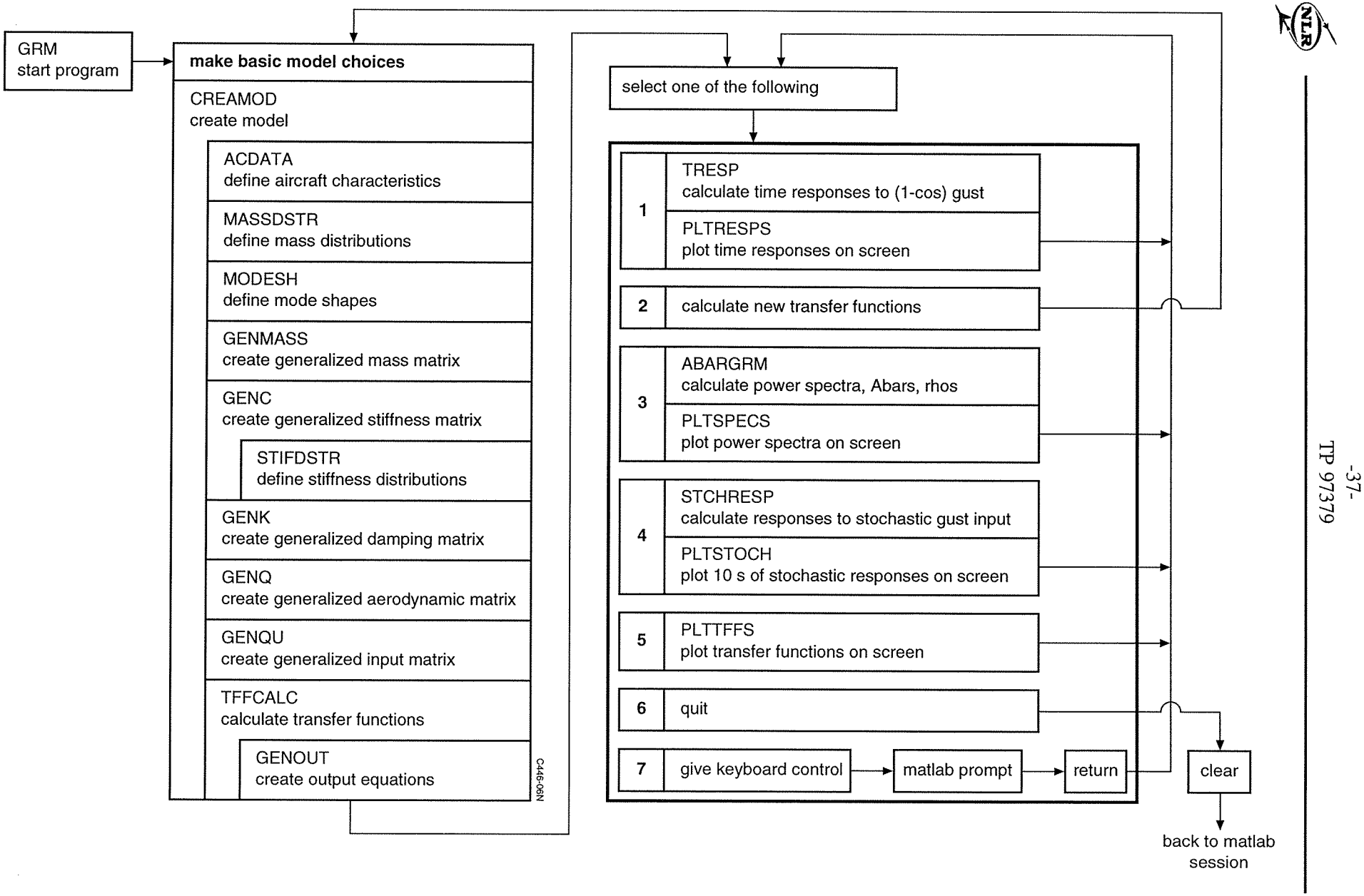


Fig. 5 Overview of gust response model program

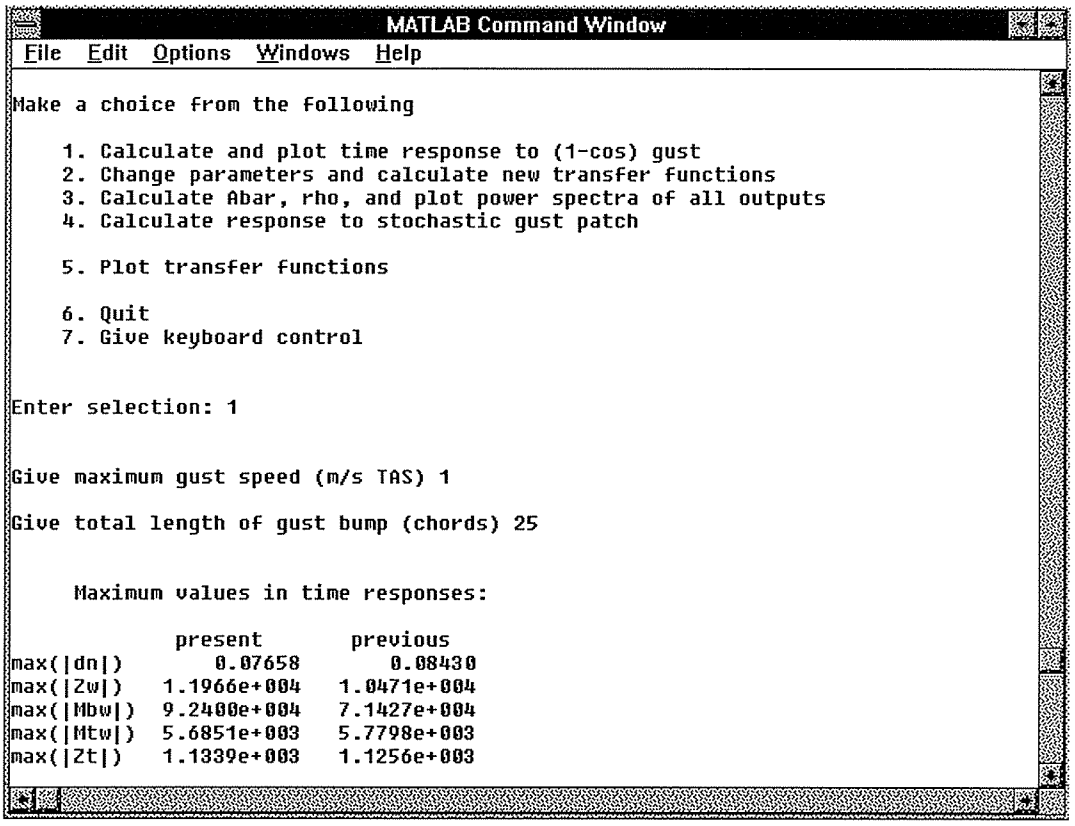


Fig. 6a Numerical output for option 1 in the menu

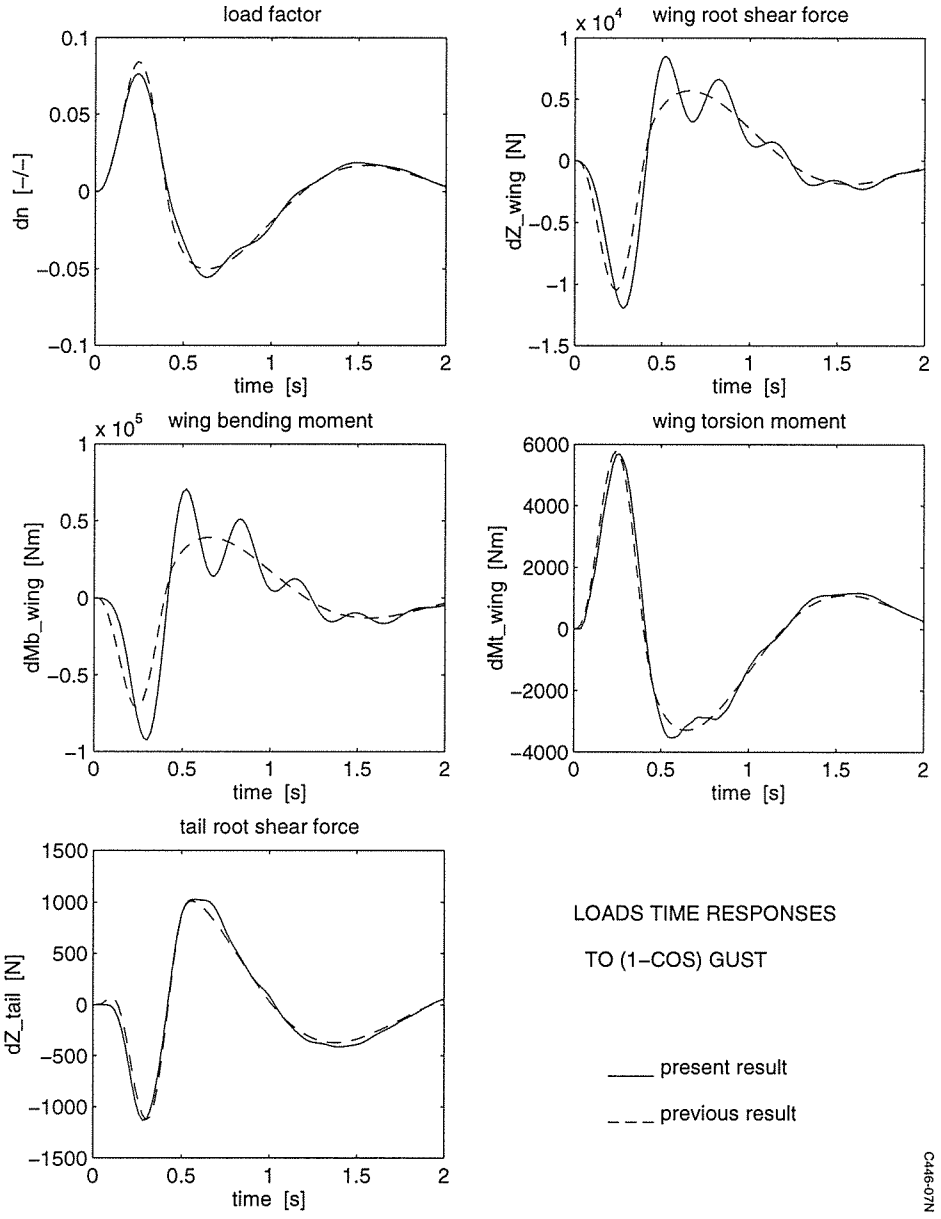


Fig. 6b Graphical output for option 1 in the menu

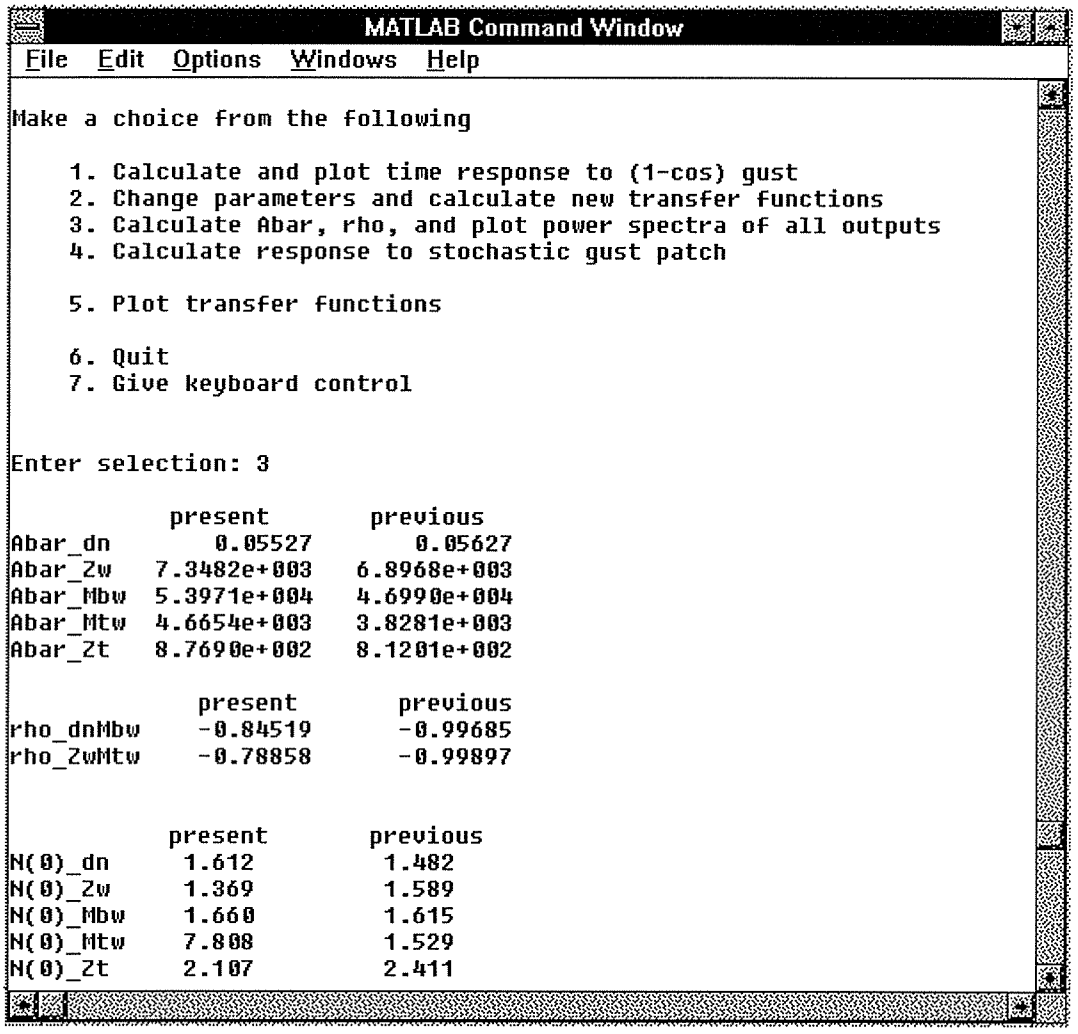


Fig. 7a Numerical output for option 3 in the menu

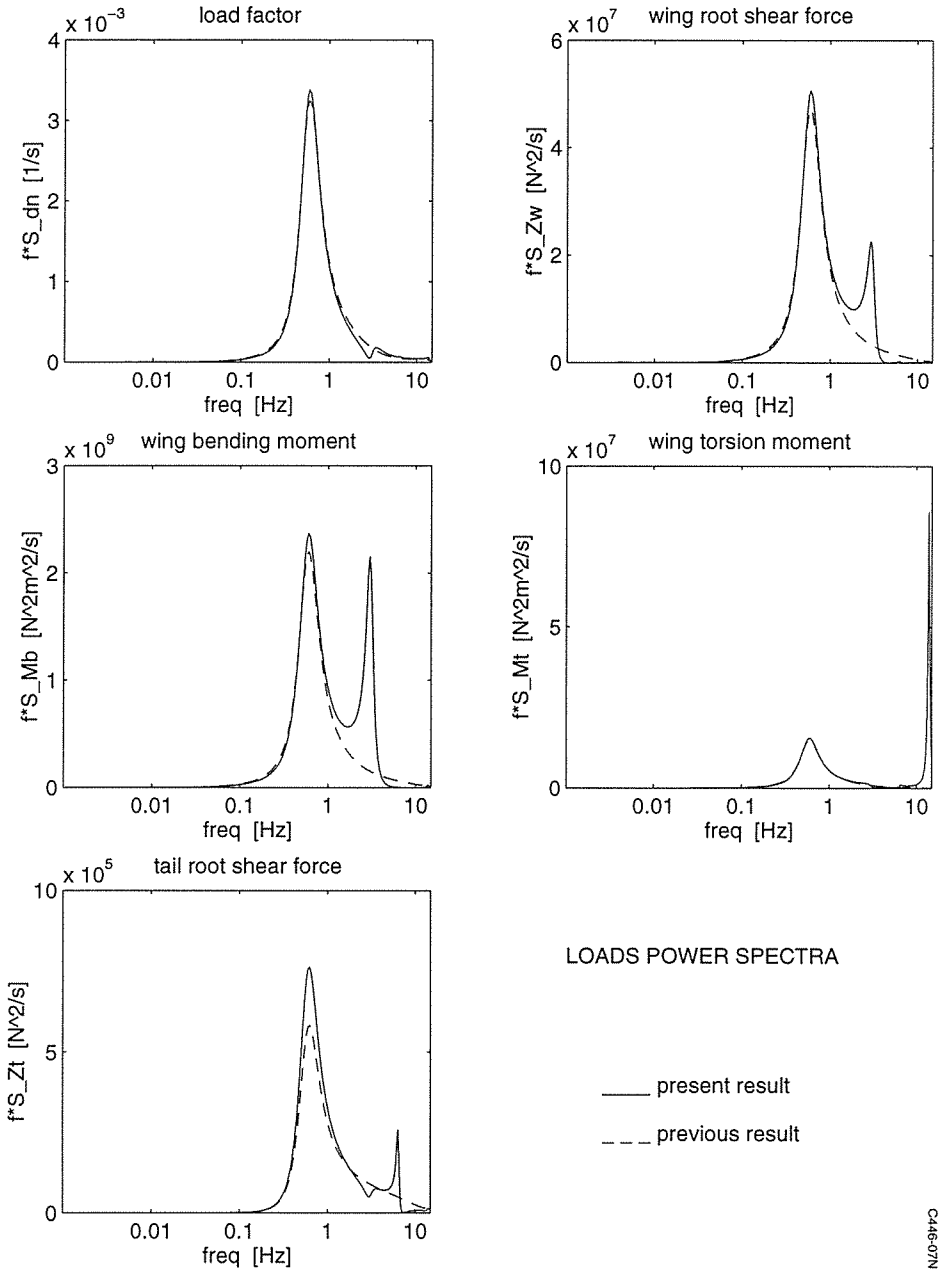


Fig. 7b Graphical output for option 3 in the menu

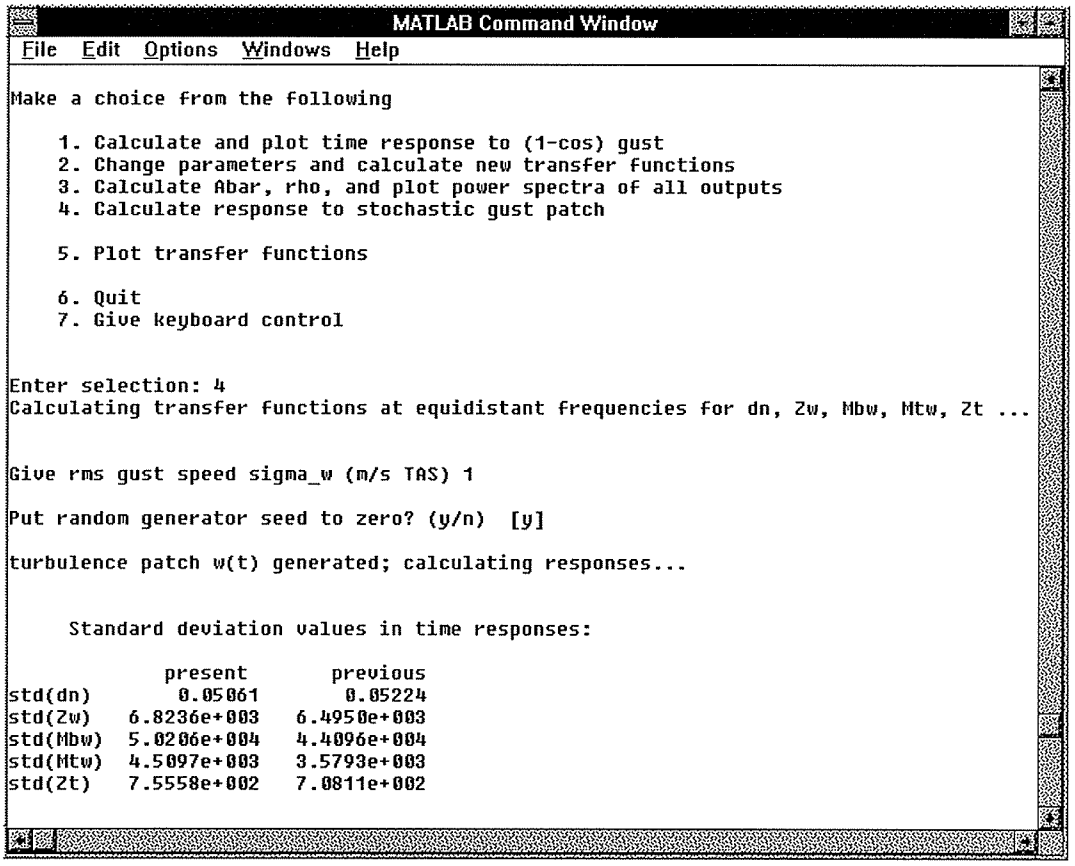


Fig. 8a Numerical output for option 4 in the menu

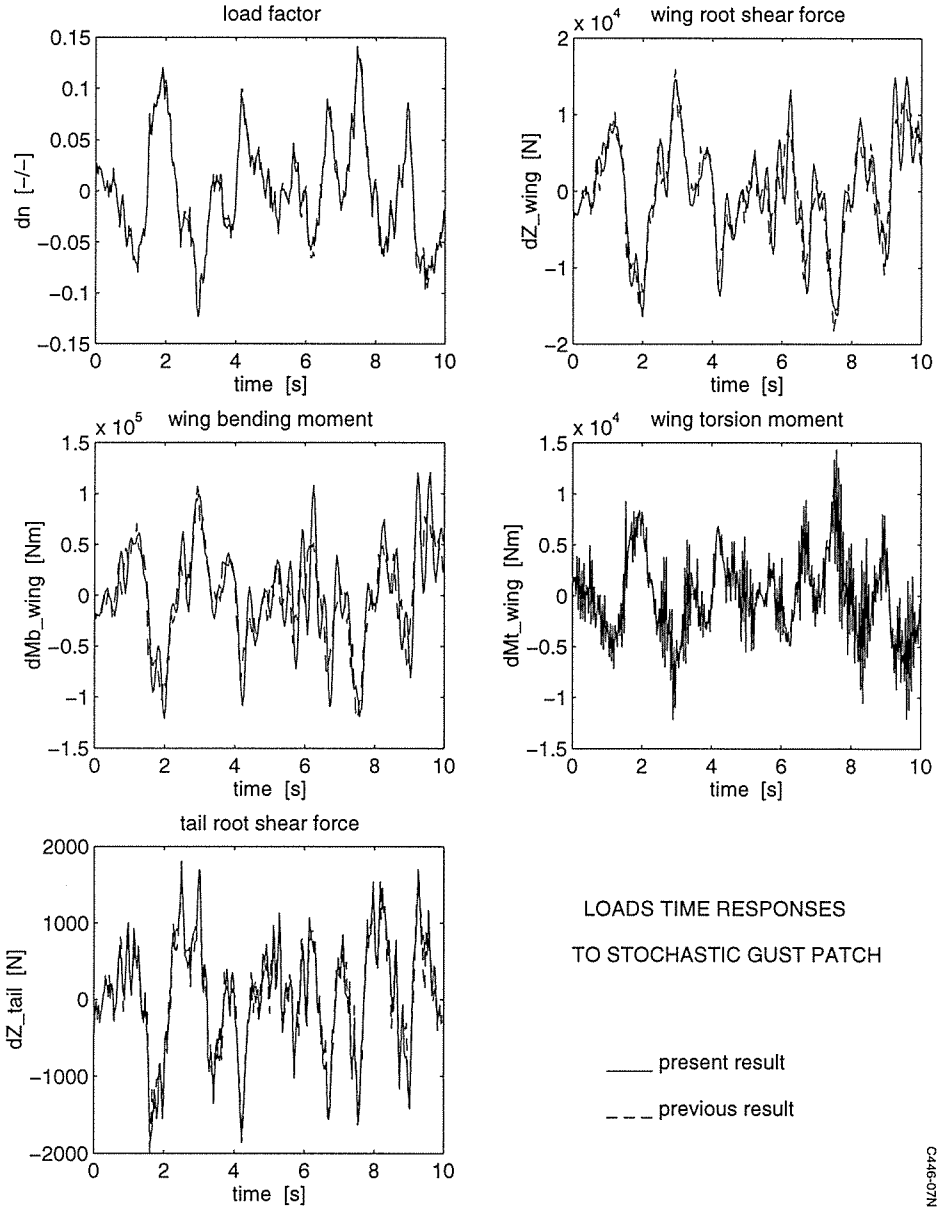


Fig. 8b Graphical output for option 4 in the menu

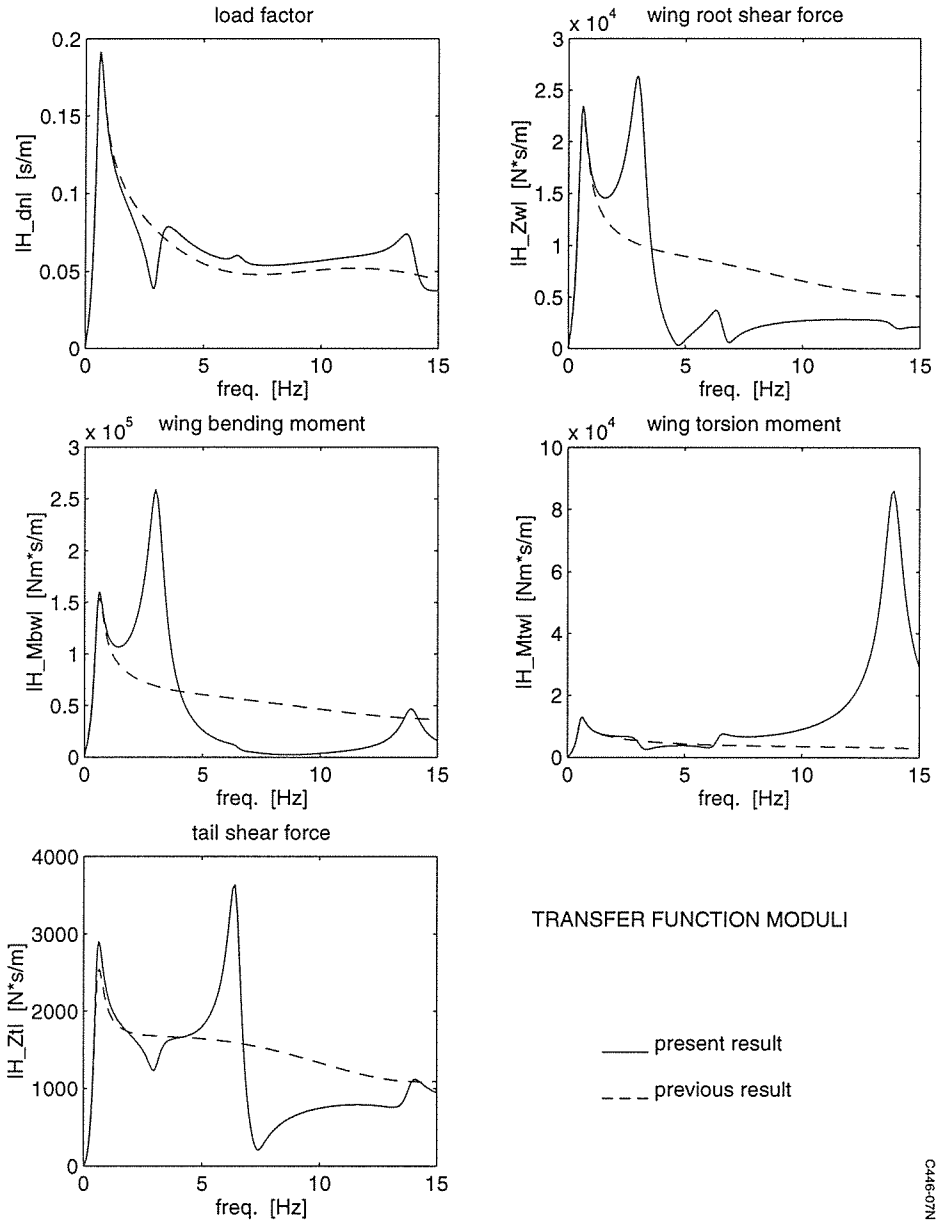
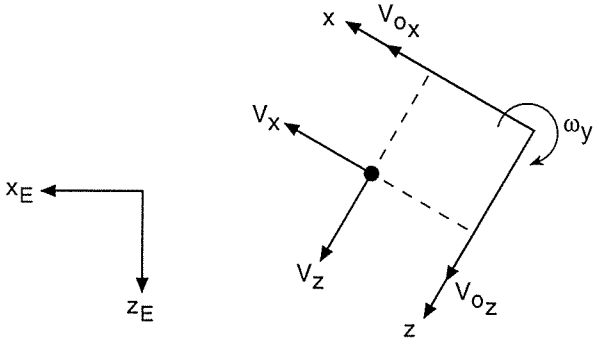


Fig. 9 Graphical output for option 5 in the menu

Appendices

A Derivation of elements in the mass- and damping matrix of the equations of motion

Consider the two-dimensional axis system (x,z) moving and rotating with respect to earth fixed axis system (x_E,z_E) with $\bar{V}_o = (V_{o_x},V_{o_z})$ and ω_y.



Velocity of a point (x,z)
with reference to (x_E,z_E):

$$V_x = V_{o_x} + \dot{x} + \omega_y z$$
$$V_z = V_{o_z} + \dot{z} - \omega_y x. \qquad \text{(A.1)}$$

We consider a "flat" = 2-dimensional aircraft hence $\bar{r} = (x,y,o)^T$, and displacements in z-direction only, hence $\dot{x} = \dot{y} = 0$.

Displacement w of point $\bar{r} = (x,y,o)^T$ in z-direction, considering n displacement modes:

$$w(x,y) = \sum_{i=1}^{n} \phi_i(x,y)\, \xi_i. \qquad \text{(A.2)}$$

Velocity of point $\bar{r} = (x,y,o)^T$:

$$V_x = V_{o_x} + \omega_y \cdot \sum_{i=1}^{n} \phi_i(x,y)\, \xi_i$$

$$V_z = V_{o_z} + \sum_{i=1}^{n} \phi_i(x,y)\, \dot{\xi}_i - \omega_y x$$

$$V_z^2 = V_{o_z}^2 + \sum_{i=1}^{n} \sum_{j=1}^{n} \phi_i(x,y)\, \phi_j(x,y)\, \dot{\xi}_i\, \dot{\xi}_j + \omega_y^2 x^2 + 2 \left[\sum_{i=1}^{n} \phi_i(x,y)\, \dot{\xi}_i \right] \cdot \left[V_{o_z} - \omega_y x \right] - 2\, V_{o_z}\, \omega_y x$$

as $V_{o_x} \gg \omega_y \sum_{i=1}^{n} \phi_i(x,y)\, \xi_i$,

$$V_x^2 \simeq V_{o_x}^2 + 2\, \omega_y\, V_{o_x} \sum_{i=1}^{n} \phi_i(x,y)\, \xi_i .$$

The kinetic energy is equal to

$$T = \int_{\text{aircraft}} \frac{1}{2} \left[V_x^2 + V_z^2 \right] dm .$$

The Lagrangian equations read:

$$\frac{d}{dt} \left[\frac{\partial T}{\partial \dot{\xi}_1} \right] - \frac{\partial T}{\partial \xi_i} + \frac{\partial U}{\partial \xi_i} = F_i .$$

We will work out the first two terms in this equation.

$$\frac{d}{dt}\left[\frac{\partial T}{\partial \dot{\xi}_i}\right] = \int_{AC} \frac{1}{2}\, dm \left[2 \sum_{j=1}^{n} \phi_i\, \phi_j\, \ddot{\xi}_j + 2\, \phi_i \left[\dot{V}_{o_z} - \dot{\omega}_y \,.\, x\right]\right]$$

$$\frac{\partial T}{\partial \xi_i} = \int_{AC} \frac{1}{2}\, dm \left[2\, \omega_y\, V_{o_x}\, \phi_i\right] \qquad \text{(A.3)}$$

$$\text{or: } \frac{d}{dt}\left[\frac{\partial T}{\partial \dot{\xi}_i}\right] - \frac{\partial T}{\partial \xi_i} = \qquad F_T(i) =$$

$$\sum_{j=1}^{n} M_{ij}\, \ddot{\xi}_j + S_i\, \dot{V}_{o_z} - R_i\, \dot{\omega}_y - S_i\, V_{o_x}\, \omega_y$$

where $M_{ij} = \int_{AC} \phi_i\, \phi_j\, dm$

$$S_i = \int_{AC} \phi_i\, dm$$

$$R_i = \int_{AC} x\, \phi_i\, dm.$$

We will work out eq. (A.3) for the rigid body modes [i=1 and 2] and for an arbitrary elastic mode ($i \geq 3$). The axis system is fixed to the aircraft in the centre of gravity, in the direction of the principal axes.

Heave: $\phi_1(x,y) = (0,0,1)^T$

Pitch: $\phi_2(x,y) = (0,0,-x)^T$

We note: $M_{11} = m$ [total ac mass]

$M_{12} = 0$

$M_{21} = 0$

$M_{22} = I_y$

$S_1 = m$

$S_2 = 0$, $S_i = M_{1i} = M_{i1}$

$R_1 = 0$, $R_i = -M_{2i} = -M_{i2}$

$T_2 = -I_y$.

Further, as the axis system x,z is fixed to the centre of gravity of the undeformed aircraft, it follows that $\xi_1=\xi_2=0$ for all t.

$$F_T(1) = \sum_{j=3}^{n} M_{1j}\,\ddot{\xi}_j + M_{11}\,\dot{V}_{o_z} - M_{11}\,V_{o_x}\,\omega_y$$

$$F_T(2) = \sum_{j=3}^{n} M_{2j}\,\ddot{\xi}_j + M_{22}\,\dot{\omega}_y$$

$$F_T(i) = \sum_{j=3}^{n} M_{ij}\,\ddot{\xi}_j + M_{i1}\,\dot{V}_{o_z} + M_{i2}\,\dot{\omega}_y - M_{i1}\,V_{o_x}\,\omega_y.$$

In matrix notation, we can write:

$$\bar{F}_T = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1n} \\ \vdots & & & \\ M_{n1} & \cdots & & M_{nn} \end{bmatrix} \begin{bmatrix} \dot{V}_{o_z} \\ \dot{\omega}_y \\ \ddot{\xi}_3 \\ \vdots \\ \ddot{\xi}_n \end{bmatrix} + \begin{bmatrix} 0 & -M_{11}V_{o_x} & 0 & \cdots & 0 \\ 0 & -M_{21}V_{o_x} & & & \\ & & & & \\ 0 & -M_{n1}V_{o_x} & & & 0 \end{bmatrix} \begin{bmatrix} V_{o_z} \\ \omega_y \\ \dot{\xi}_3 \\ \vdots \\ \dot{\xi}_n \end{bmatrix}.$$

Or

$$\bar{F}_T(\bar{\xi}) = M\,\ddot{\bar{\xi}} + D\,\dot{\bar{\xi}}$$

where $\dot{\bar{\xi}} = (V_{o_z}, \omega_y, \dot{\xi}_3, \ldots \dot{\xi}_n)^T$.

The elements M_{ij} of M are defined as $M_{ij} = \int \phi_i\,\phi_j\,dm$, or, in matrix notation for an aircraft consisting of k discrete elements

$$M_{ij} = \sum_{k} \bar{\phi}_{i,k}^{T}\,[m]_k\,\bar{\phi}_{j,k}.$$

The elements D_{ij} of D are defined as:

$D_{ij} = 0$ for $j \neq 2$

$D_{i2} = -V_{o_x} M_{i1}$

or:

$D_{ij} = 0$ for $j \neq 2$

$$D_{i2} = -V_{o_x} \sum_k \bar{\phi}_{i,k}^T [m]_k \bar{\phi}_{1,k} \ .$$

B Matlab files of Gust Response Model

```
% GRM.m
%
% version 1.0
% 7 July 1997
%-----------------------------------
%
% Main program for the aircraft Gust Response Model

clear


prevt=[]; prevm=[]; prevS=[]; prevA=[]; prevH=[]; prevstoch=[]; prevstd=[];

ncr = 1;
while ncr==1

 modesw=0;
 while (modesw~=1 & modesw~=2 & modesw~=3)
  disp(' ');
  disp('Available types of aircraft model:');
  disp('  1 = plunge only');
  disp('  2 = plunge + pitch');
  disp('  3 = plunge + pitch +  3 flexible modes');
  disp(' ');
  modesw=input('Enter the number of your choice: ');

  if (modesw~=1 & modesw~=2 & modesw~=3)
   disp('WRONG KEYBOARD INPUT: Degrees of freedom choice must be 1, 2, or 3');
  end
 end

 disp(' ');
 aerinsw=input('Account for aerodynamic inertia? (y/n) [y]','s');
 disp(' ');

    % create aircraft Gust Response Model (calculate transfer functions)

  creamod

  ncal= 1;
  texist=0;
  specexist=0;
  stochexist=0;
  while ncal==1

   disp(' ');
   disp('Make a choice from the following');
```

```
disp(' ');
disp('    1. Calculate and plot time response to (1-cos) gust');
disp('    2. Change parameters and calculate new transfer functions');
disp('    3. Calculate Abar, rho, and plot power spectra of all outputs');
disp('    4. Calculate response to stochastic gust patch');
disp(' ');
disp('    5. Plot transfer functions');
disp(' ');
disp('    6. Quit');
disp('    7. Give keyboard control');

disp(' ');
ncont = input('Enter selection: ');


if ncont == 1
  if texist==1
    prevt = [t dnt dzwt dmbwt dmtwt dztt wt];
    prevm = max([abs(dnt) abs(dzwt) abs(dmbwt) abs(dmtwt) abs(dztt) abs(wt)]);
  end
  tresp
  texist=1;
  pltresps
elseif ncont == 2
  ncal=0;
elseif ncont == 3
  AbarGRM
  specexist=1;
  pltspecs
elseif ncont == 4
  if stochexist==1
    prevstoch = [tst dntst dzwtst dmbwtst dmtwtst dzttst wtst];
    prevstd   = std(prevstoch(:,2:7));
  end
  stchresp
  stochexist=1;
  pltstoch
elseif ncont == 5
  plttffs
elseif  ncont == 6
  ncr =0;
  ncal=0;
elseif ncont == 7
  disp('type r e t u r n  to get back to the menu')
  keyboard
end

end
if texist==1
 prevt = [t dnt dzwt dmbwt dmtwt dztt wt];
```

```
    prevm = max([abs(dnt) abs(dzwt) abs(dmbwt) abs(dmtwt) abs(dztt) abs(wt)]);
  else
    prevt = [];  prevm = [];
  end
  prevH = [f tff];
  if specexist==1
    prevS = [f Syy];
    prevA = [Abars Nnul];
    prevrdnmbw = rhodnmbw;
    prevrzwmtw = rhozwmtw;
  else
    prevS = [];  prevA = [];
  end
  if stochexist==1
    prevstoch = [tst dntst dzwtst dmbwtst dmtwtst dzttst wtst];
    prevstd   = std(prevstoch(:,2:7));
  else
    prevstoch = [];
    prevstd   = [];
  end

end

clear



% creamod.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Creates the aircraft model equations of motion

acdata
massdstr

modesh
genmass
genC
genK
genQ
genQu

% Matrices are now defined for the following equations of motion:
%       GM*x*s^2 + GK*x*s - GQ*x + GC*x = GQu*u

% transfer functions of outputs will now be calculated at specified frequencies
tffcalc
```

```
% acdata.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Default and fixed values for aircraft characteristics
% MODIFICATION d.d. 29-5-1997: structural damping default added  (WJV)


cw    = 3.83;     % wing chord (= mean aerodynamic chord)
b     = 24;       % wing span
labda = 17*pi/180;% sweepback angle
ew    = .35;      % elastic axis position in wing chord lengths

ct    = 2.29;     % stabilizer chord
bt    = 10;       % stabilizer span
xt    = 17;       % distance between stabilizer e.a. and Origin in nose of mac
et    = .25;      % elastic axis position in tail chord lengths

cxcg  = .15;      % centre of gravity position in m.a.c.
m     = 20e3;     % half aircraft mass
Iy    = .8122e6;  % half aircraft moment of inertia around lateral axis
gs    = .03;      % amount of structural damping

czaw  = -6.379;   % wing strips lift coefficient
czat  = -4.61;    % stabilizer strips lift coefficient
deda  = .35;      % downwash coefficient
cmaf  = .4;       % fuselage moment coefficient

V     = 220;      % speed in m/s TAS
rho   = .59;      % air density (this default corresponds with 7000 m)



% The following multiplication factors can be applied

% TO MASS AND INERTIA DISTRIBUTIONS OF WING, REAR FUSELAGE, AND
STABILIZER:
mumw=1;  mumt=1;    mumf=1;
muIxw=1; muIyw=1;   muIyt=1; muIyf=1;
mumtot=1;

% TO GENERALIZED STIFFNESSES OF THE THREE ELASTIC MODES:
muC33=1; muC44=1; muC55=1;



disp(' ');
```

```
disp('The default aircraft model characteristics are as follows:');
disp(' ');
disp('cw    = 3.83;     % wing chord (= mean aerodynamic chord)');
disp('b     = 24;       % wing span');
disp('Sw    = b*cw;     % wing surface area (used only for fuselage Y-moment)');
disp('labda = 17*pi/180;% sweepback angle');
disp('ew    = .35;      % elastic axis position in wing chord lengths from nose');
disp(' ');
disp('ct    = 2.29;     % stabilizer chord');
disp('bt    = 10;       % stabilizer span');
disp('xt    = 17;       % distance between stabilizer e.a. and Origin in nose of mac');
disp('et    = .25;      % elastic axis position in tail chord lengths from nose');
disp(' ');
disp('cxcg  = .15;      % centre of gravity position in m.a.c.');
disp('m     = 20e3;     % half aircraft mass');
disp('Iy    = .8122e6;  % half aircraft moment of inertia around lateral axis');
disp('gs    = .03       % amount of structural damping');
disp(' ');
disp('czaw  = -6.379;   % wing strips aerodynamic Z force coefficient');
disp('czat  = -4.61;    % stabilizer strips aerodynamic Z force coefficient');
disp('deda  = .35;      % downwash coefficient');
disp('cmaf  = .4;       % fuselage moment coefficient');
disp(' ');
disp('V     = 220;      % speed in m/s TAS');
disp('rho   = .59;      % air density (this default corresponds with 7000 m)');
disp(' ');

change=input('Change any of these values? (y/n) [n]','s');
disp(' ');
if change=='y'
  disp('Give new values of variables mentioned above');
  disp('when ready, type the letters r e t u r n and give <ENTER>');
  disp(' ');
  keyboard
end

clc

Sw    = b*cw;     % ~ 92 m^2 by default (this relation cannot be modified)
xcg   = -cxcg*cw; % centre of gravity position
lt    = xt+xcg;   % distance between stabilizer e.a. and centre of gravity

xref  = b/4*tan(labda)-ew*cw;     % reference point for wing loads is
                                  % intersection of a/c centerline and wing
                                  % elastic axis. (for wing torsion calculation)
span  = b;
b     = span/cos(labda);          % this b is defined along the wing el. axis
cZaw  = czaw*cos(labda);
cZat  = czat;
cMaf  = cmaf;
```

```
disp(' ');
disp('Default mass- and inertia distributions of wing, fuselage, and tail');
disp('are multiplied by the factors:');
disp(' ');
disp('mumw  = 1;      % for wing mass distribution');
disp('muIxw = 1;      % for wing inertia around elastic axis distribution');
disp('muIyw = 1;      % for wing inertia around local y distribution');
disp('mumt  = 1;      % for stabilizer mass distribution');
disp('muIyt = 1;      % for stabilizer inertia around local y distribution');
disp('mumf  = 1;      % for rear fuselage mass distribution');
disp('muIyf = 1;      % for rear fuselage inertia around local y distribution');
disp(' ');
disp('mumtot= 1;      % overall factor applied to ALL local masses and inertia');
disp(' ');
disp('These element distributions are used in the generation of generalized');
disp('mass matrix flexible mode elements and in loads calculations');
disp(' ');
change=input('Change any of these values? (y/n) [n]','s');
disp(' ');
if change=='y'
   disp('Give new values of variables mentioned above');
   disp('when ready, type the letters r e t u r n and give <ENTER>');
   disp(' ');
   keyboard
end

clc

if modesw==3
  disp(' ')
  disp('Elements on the stiffness matrix diagonal are multiplied by:')
  disp(' ')
  disp('        muC33=1; muC44=1; muC55=1;')
  disp(' ')

  change=input('Change any of these values? (y/n) [n]','s');
  disp(' ');
  if change=='y'
     disp('Give new values of variables mentioned above');
     disp('when ready, type the letters r e t u r n and give <ENTER>');
     disp(' ');
     keyboard
  end
end

clc
disp('Calculating transfer functions for dn, Zw, Mbw, Mtw, Zt ...');
```

```
% massdstr.m
%
% version 1.0
% 7 July 1997
% version 1.1
% 6 October 1997
%-------------------------------------
%
% set default distributions for masses and inertia of wing, fuselage
% and tail, inspired by Fokker 100 data
% MODIFICATION d.d. 6-10-1997: wing strip width given in Y-direction (WJV)

% WING - 5 strips with mass and inertia

%bsw=b/2/5;                    % width of wing strips
bsw=span/2/5;                  % width of wing strips  MODIFICATION d.d. 6-10-1997
xwl=[b/2/5/2:b/2/5:b/2-b/2/5/2]';   % local x' coordinates of wing strips

distr=[1 4/5 3/5 2/5 1/5]';
mw  = mumtot * mumw   * distr*6000/sum(distr);
Ixwl= mumtot * muIxw * distr*3100/sum(distr);
Iywl= mumtot * muIyw * distr*(2.0214e5-mw'*xwl.^2)/sum(distr);

% TAIL - 1 strip with mass and inertia

bst=bt/2;
xtl=2*bst/5;                   % there is only one tail strip
mt  = mumtot * mumt   * 290;
Ixtl= mumtot * 135;
Iytl= mumtot * muIyt * (1.6281e3-mt*xtl^2);

% REAR FUSELAGE - mass and inertia lumps

xfl=xt/17*(.4*3.83+[1.090 3. 3.962 6.062 7.183 7.837 9.591 11.6 13.43 14.4]');
                        % origin at 0% mac ;   at 14.4 m is vert. tail

mf = mumtot * mumf *  [891.6  187.5  681.5  731.1  218.7  418.9  101.1  88.74  89.24 230]';
Iyf= mumtot * muIyf*1e3*[1.4470 0.0262 0.5013 0.6634 0.3649 0.4216 0.1309 0.0701 0.0747
3]';
                        % the vertical tail Iy is high because of the
                        % z-coordinate


% modeshapes.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Define 5 mode shapes: plunge, pitch, rear fuselage bending,
```

```
%                       wing bending, and wing torsion
% There is a total of 16 mass points in this aircraft model
%
% MODIFICATION d.d. 23-5-1997: 4th order wing bending mode included  (WJV)
% MODIFICATION d.d. 29-5-1997: 2nd order wing torsion mode included  (WJV)

clear PHIw PHIpsi PHIthe

xfg=-xfl;                   % local fuselage co-ordinates xfl from massdistr.m
xwg=-(xwl-b/4)*sin(labda)-ew*cw;          % Origin is in mac-nose

% PLUNGE

PHIw(:,1)=ones(16,1);
PHIpsi(:,1)=zeros(16,1);
PHIthe(:,1)=zeros(16,1);

if (modesw==2 | modesw==3)

% PITCH

PHIw(1:10,2)=-(xfg-xcg)/lt;
PHIw(11:15,2)=-(xwg-xcg)/lt;
PHIw(16,2)=1;
PHIpsi(:,2)=zeros(16,1);
PHIthe(:,2)=ones(16,1)/lt;

end

if modesw==3

% REAR FUSELAGE BENDING

PHIw(1:10,3)=0.5*((xfl/xt).^2).*(3-xfl/xt);    % Bending starts at Origin
                    % This can introduce a discontinuity at the wing/
                    % fuselage intersection, if this point is behind
                    % the Origin. (An aesthetic problem only)
PHIw(11:15,3)=zeros(5,1);
PHIw(16,3)=1;
PHIpsi(:,3)=zeros(16,1);
PHIthe(1:10,3)=3/xt*(xfl/xt-0.5*((xfl/xt).^2));
PHIthe(11:15,3)=zeros(5,1);
PHIthe(16,3)=3/2/xt;

% WING BENDING

%PHIw(1:10,4)=zeros(10,1);
%PHIw(11:15,4)=(xwl*2/b).^2;
%PHIw(16,4)=0;
%PHIpsi(1:10,4)=zeros(10,1);
```

```
%PHIpsi(11:15,4)=-8*xwl/b^2*sin(3/2*pi-labda);
%PHIpsi(16,4)=0;
%PHIthe(1:10,4)=zeros(10,1);
%PHIthe(11:15,4)=-8*xwl/b^2*cos(3/2*pi-labda);
%PHIthe(16,4)=0;
    % 4th order wing bending:
PHIw(1:10,4)=zeros(10,1);
PHIw(11:15,4)=1/3*( (1-xwl*2/b).^4 - 4*(1-xwl*2/b) + 3 );
PHIw(16,4)=0;
PHIpsi(1:10,4)=zeros(10,1);
PHIpsi(11:15,4)=-8/3/b*( 1 - (1-xwl*2/b).^3 )*sin(3/2*pi-labda);
PHIpsi(16,4)=0;
PHIthe(1:10,4)=zeros(10,1);
PHIthe(11:15,4)=-8/3/b*( 1 - (1-xwl*2/b).^3 )*cos(3/2*pi-labda);
PHIthe(16,4)=0;

% WING TORSION

PHIw(:,5)=zeros(16,1);
PHIpsi(1:10,5)=zeros(10,1);
PHIpsi(11:15,5)=( -4/b^2*xwl.^2 + 4/b*xwl )*cos(3/2*pi-labda)/cw;
                                    % divide by cw for correct units
PHIpsi(16,5)=0;                         % in state vector ([m])
PHIthe(1:10,5)=zeros(10,1);
PHIthe(11:15,5)=-1*( -4/b^2*xwl.^2 + 4/b*xwl )*sin(3/2*pi-labda)/cw;
                                    % divide by cw for correct units
PHIthe(16,5)=0;                         % in state vector

end
```

```
% genmass.m
%
% version 1.0
% 7 July 1997
%-----------------------------------
%
% Create generalized mass matrix from mode shapes and local mass data.
% The rigid modes plunge and pitch generalized masses are forced to be
% equal to the aircraft mass and moment of inertia respectively.

% RIGID MODES

gm=[m 0 ; 0 Iy/lt^2];       % pitch mode has been normalized by 1/lt


% TRANSFORM LOCAL INERTIA VALUES TO GLOBAL AXES
% (these values are also used in output calculations)
```

```
phi=3/2*pi-labda;          % rotation angle between local wing axes and
                        % global aircraft axes
Ixwg = (cos(phi)^2)*Ixwl + (sin(phi)^2)*Iywl ;  % Ixyl is assumed to be zero; so
Iywg = (sin(phi)^2)*Ixwl + (cos(phi)^2)*Iywl ;  % local axes are principal axes
Ixywg= sin(phi)*cos(phi)*(Ixwl-Iywl);

Ixtg = Iytl;
Iytg = Ixtl;


if modesw==3

% FLEXIBLE MODES

  Mw = diag([mf;mw;mt]);      % inertia data for z-DOF in each point

  Mpsi    = diag([zeros(10,1) ; Ixwg ; Ixtg]);
  Mthe    = diag([Iyf ; Iywg ; Iytg]);
  Mpsithe = diag([zeros(10,1) ; Ixywg ; 0]);

  GM = PHIw'*Mw*PHIw + PHIpsi'*Mpsi*PHIpsi + PHIthe'*Mthe*PHIthe -...
       PHIpsi'*Mpsithe*PHIthe - PHIthe'*Mpsithe*PHIpsi;
  GM(1:2,1:2)=gm;

elseif modesw==2

  GM=gm;

else

  GM=gm(1,1);

end
```

```
% genC.m
%
% version 1.0
% 7 July 1997
%-----------------------------------
%
% Create generalized stiffness matrix from stiffnes distributions.
% Rigid modes stiffness values are zero.
%
% MODIFICATION d.d. 27-5-1997: 4th order wing bending mode and 3rd order
%                              beam element displacement functions included  (WJV)
% MODIFICATION d.d. 29-5-1997: 2nd order wing torsion mode included          (WJV)
%                              structural damping gs added                   (WJV)
```

```
if modesw==3

stifdstr          % local stiffness distribution over elements

% local co-ordinates of left and right nodes of each element

xfl_l     = [0 ; ( xfl(1:9)+xfl(2:10) )/2];
xfl_r     = ( xfl+[xfl(2:10);xt] )/2;

xwl_l     = [0:b/2/5:b/2-b/2/5]';
xwl_r     = [b/2/5:b/2/5:b/2]';

% calculate derivatives of deflection angles

thederf_l = [zeros(10,2) 3/xt^2*(1-xfl_l/xt) zeros(10,2)];      % w"_l rear fuselage
thederf_r = [zeros(10,2) 3/xt^2*(1-xfl_r/xt) zeros(10,2)];      % w"_r rear fuselage

psiderw   = [zeros(5,4)  (-8/b^2*xwl + 4/b)/cw];    % psi' is linear for wing torsion

thederw_l = [zeros(5,3) (16/b^2)*(1-xwl_l*2/b).^2   zeros(5,1)];
thederw_r = [zeros(5,3) (16/b^2)*(1-xwl_r*2/b).^2   zeros(5,1)];


bsf       = xfl_r - xfl_l;
EIdxf     = diag(EIyf)*diag(bsf);
EIdxw     = diag(EIyw)*b/10*eye(5);
GJdxw     = diag(GJw)*b/10*eye(5);

GCfb = 1/6*( 2*thederf_l'*EIdxf*thederf_l + 2*thederf_r'*EIdxf*thederf_r + ...
           thederf_l'*EIdxf*thederf_r + thederf_r'*EIdxf*thederf_l );
GCwb = 1/6*( 2*thederw_l'*EIdxw*thederw_l + 2*thederw_r'*EIdxw*thederw_r + ...
           thederw_l'*EIdxw*thederw_r + thederw_r'*EIdxw*thederw_l );
GCwt = psiderw'*GJdxw*psiderw;

GC = GCfb + GCwb + GCwt;


GC(3,3)= (muC33+j*gs)*GC(3,3);          % gs is the structural damping
GC(4,4)= (muC44+j*gs)*GC(4,4);
GC(5,5)= (muC55+j*gs)*GC(5,5);


elseif modesw==2

  GC=diag([0; 0]);

else
```

```
  GC=0;

end


% stifdstr.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% set distributions for stiffnesses of wing, fuselage
% and tail, inspired by Fokker 100 data

% WING - 5 mass points, 5 elements

EIyw = [16.9e+07 9.52e+07 3.45e+07 1.21e+07 4.90e+06]';
GJw  = [12.8e+07 6.48e+07 2.28e+07 8.10e+06 3.30e+06]';


% TAIL

% connection between fuselage and stabilizer remains straight (no My).


% REAR FUSELAGE - 10 mass points, 10 elements

EIyf= [13.2e8 9.95e8 8.55e8 7.95e8 7.35e8 5.75e8 4.70e8 3.5e8 2.4e8 .95e8]';



% genK.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Creates the K-matrix, containing the contributions from rotation of the
% aircraft axis system (omega).
% MODIFICATION d.d. 29-5-1997: additional terms in damping matrix due to omega   (WJV)

if modesw == 3
  GK=zeros(5,5);
  GK(1,2)=-V*m*PHIthe(1,2)*PHIw(1,1);

  GK(3:5,2) = -V*PHIthe(1,2)*PHIw(:,3:5)'*[mf;mw;mt];   % the additional terms

elseif modesw ==2
  GK=zeros(2,2);
```

```
  GK(1,2)=-V*m*PHIthe(1,2)*PHIw(1,1);
else
  GK=0;
end
```

```
% genQ.m
%
% version 1.0
% 7 July 1997
%-----------------------------------
%
% Create generalized aerodynamic matrices for strip model
% NOTE: for wing strips lift at .25cw,  no moment,  alfa at .75cw
%       for tail strips lift at .25ct,  no moment,  alfa at .75ct
%       for fuselage only moment around Ygl due to plunge mode
% MODIFICATION d.d. 29-5-1997: aerodynamic moment due to panel rotation added  (WJV)

dp = 0.5*rho*V*V;            % dynamic pressure


% ANGLES OF INCIDENCE OF ALL STRIPS

alfawQ1 = PHIw(11:15,:)/V + PHIthe(11:15,:)*(.75-ew)*cw/V;
alfatQ1 = PHIw(16,:)/V + PHIthe(16,:)*(.75-et)*ct/V;

if modesw == 3
  alfawQ0 = [zeros(5,2) , PHIthe(11:15,3:5)];
  alfatQ0 = [zeros(1,2) , PHIthe(16,3:5)];
  alfafQ1 = zeros(1,5);
elseif modesw == 2
  alfawQ0 = [zeros(5,2)];
  alfatQ0 = [zeros(1,2)];
  alfafQ1 = zeros(1,2);
else
  alfawQ0 = [zeros(5,1)];
  alfatQ0 = 0;
  alfafQ1 = 0;
end

alfafQ1(1,1) = PHIw(1,1)/V;  % fuselage is one strip with surface area Sw/2
                             % only plunge mode causes alfa_fuselage


% AERO FORCES ON STRIPS

Z1w = dp*cw*bsw*cZaw*alfawQ1;
M1w = -dp*cw*bsw*cZaw*( alfawQ1*(ew-.25)*cw-cw/16*cw/V*PHIthe(11:15,:) );
Z0w = dp*cw*bsw*cZaw*alfawQ0;          % M1w contains the contribution due to rotation
```

```
                                        % of the panel, as given by Ref. FUNG.
Z1t = dp*ct*bst*cZat*alfatQ1;
Z0t = dp*ct*bst*cZat*alfatQ0;

M1f = dp*cw*b/2*cMaf*alfafQ1;


% CONTRIBUTIONS TO GENERALIZED AERO MATRIX FROM EACH PART

Q1w = PHIw(11:15,:)'*Z1w + PHIthe(11:15,:)'*M1w;
Q0w = PHIw(11:15,:)'*Z0w + PHIthe(11:15,:)'*(-Z0w*(ew-.25)*cw);

Q1t = PHIw(16,:)'*Z1t + PHIthe(16,:)'*(-Z1t*(et-.25)*ct);
Q0t = PHIw(16,:)'*Z0t + PHIthe(16,:)'*(-Z0t*(et-.25)*ct);

if modesw==1
  Q1f = 0;       % Mf only influences the pitching mode
  Q0f = 0;
elseif modesw==2
  Q1f = [0 PHIthe(1,2)]'*M1f;    % Mf only influences the pitching mode
  Q0f = zeros(2,2);
else
  Q1f = [0 PHIthe(1,2) 0 0 0]'*M1f;    % Mf only influences the pitching mode
  Q0f = zeros(5,5);
end


% DOWNWASH CONTRIBUTION   downwash is calculated with respect to 2nd wing strip

downw1=alfawQ1(2,:)*deda;
downw0=alfawQ0(2,:)*deda;

tau=(xt+xwg(2))/V;          % time delay between 2nd wing strip and tail strip

Z1d = dp*ct*bst*cZat*(-downw1);
Z0d = dp*ct*bst*cZat*(-downw0);

Q1d = PHIw(16,:)'*Z1d + PHIthe(16,:)'*(-Z1d*(et-.25)*ct);
Q0d = PHIw(16,:)'*Z0d + PHIthe(16,:)'*(-Z0d*(et-.25)*ct);


% THEODORSEN FUNCTIONS

if aerinsw == 'n'
  theows = 'theow=1;';
  theots = 'theot=1;';
else
  vcw = V/cw;  vct = V/ct;
  theows = 'theow=(.5*s*s+.56085*s*vcw+.054*vcw*vcw)/((s+.09*vcw)*(s+.6*vcw));';
  theots = 'theot=(.5*s*s+.56085*s*vct+.054*vct*vct)/((s+.09*vct)*(s+.6*vct));';
```

```
end


% TOTAL GENERALIZED AERODYNAMIC MATRIX

GQs = ['GQ = s*(Q1w+Q1f)*theow + s*Q1t*theot '...
       '+ (Q0w+Q0f)*theow + Q0t*theot + s*Q1d*exp(-tau*s)*theot'...
       '+ Q0d*exp(-tau*s)*theot;'];
                                                % NOTE that this is
                                                % just a string!
% These strings are evaluated later on, in tffcalc.m, when s is known.



% genQu.m
%
% version 1.0
% 7 July 1997
%-----------------------------------
%
% Create generalized input matrix for vertical gust input

% ANGLES OF INCIDENCE OF ALL STRIPS DUE TO VERTICAL GUST SPEED

alfawu = ones(5,1)/V;
alfatu = 1/V;
alfafu = 1/V;


% GENERALIZED FORCES

Zwu = dp*cw*bsw*cZaw*alfawu;
Ztu = dp*ct*bst*cZat*alfatu;
Mfu = dp*cw*b/2*cMaf*alfafu;


% TIME DELAYS WITH RESPECT TO FIRST WING STRIP (no delay applied to fuselage)
tdelw = -(xwg-xwg(1))/V;
tdelt = (xt+xwg(1))/V;


% CONTRIBUTIONS TO GENERALIZED INPUT MATRIX FROM EACH PART

Quw = ['( PHIw(11:15,:)''*(Zwu.*exp(-tdelw*s)) + '...
       'PHIthe(11:15,:)''*((-Zwu.*exp(-tdelw*s))*(ew-.25)*cw) )'];
Qut = ['( PHIw(16,:)''*(Ztu.*exp(-tdelt*s)) + '...
       'PHIthe(16,:)''*(-(Ztu.*exp(-tdelt*s))*(et-.25)*ct) )'];
if modesw == 3
  Quf = [0 PHIthe(1,2) 0 0 0]'*Mfu;
elseif modesw == 2
```

```
  Quf = [0 PHIthe(1,2)]'*Mfu;
else
  Quf =0;
end


% DOWNWASH CONTRIBUTION   downwash is calculated with respect to 2nd wing strip

downwu=alfawu(2,1)*deda;
Zdu = dp*ct*bst*cZat*(-downwu);
Qud = PHIw(16,:)'*Zdu + PHIthe(16,:)'*(-Zdu*(et-.25)*ct);


% SEARS FUNCTIONS

if aerinsw == 'n'
  searws = 'searw=1;';
  searts = 'seart=1;';
else
  searws = 'searw=(1.13*s*vcw+.52*vcw*vcw)/((s+.26*vcw)*(s+2*vcw));';
  searts = 'seart=(1.13*s*vct+.52*vct*vct)/((s+.26*vct)*(s+2*vct));';
end


% TOTAL GENERALIZED AERODYNAMIC MATRIX FOR GUST INPUTS (s=j*omega
must be known)

GQus=['GQu=' Quw '*searw +' Qut '*seart + Quf*searw + Qud*exp(-tau*s);'];



% genout.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Create output matrices for outputs:
%          d_n, dZ_wing, dMb_wing, dMt_wing, and dZ_tail

% d_n=(-s^2 * h + s * theta * V -
%                      - s^2 * 1/m_ac * sum(m_k*w_k3+m_k*w_k4+m_k*w_k5)/9.81
%
% fuselage and wing bending cause a cg displacement (modes are not orthogonal
% to the plunge mode)

if modesw == 3
  dnC = [-s^2*PHIw(1,1) s*V*PHIthe(1,2) (-s^2)/m*diag(Mw)'*PHIw(:,3:5)]/9.81;
elseif modesw == 2
  dnC = [-s^2*PHIw(1,1) s*V*PHIthe(1,2)]/9.81;
```

```
else
  dnC = [-s^2*PHIw(1,1)]/9.81;
end

dnD = 0;

% dZ_wing  wing root shear force  (root is at a/c centerline), pos. down

dZwC = (s*sum(Z1w) + sum(Z0w)) * theow;
dZwCi= -s^2 * sum(diag(mw)*PHIw(11:15,:));
if modesw ~= 1
  dZwCi(1,2)=dZwCi(1,2)+s*V*PHIthe(1,2)*sum(mw);
end
dZwC = dZwC+dZwCi;
dZwD = sum(Zwu.*exp(-tdelw*s))*searw;


% dMx_wing  wing root "bending" moment  (root is at a/c centerline),
% pos. tip down

dMxwC = (s*sum(diag(xwl*cos(labda))*Z1w) + sum(diag(xwl*cos(labda))*Z0w))*...
        theow;
dMxwCi= -s^2 * sum( diag(mw.*xwl*cos(labda))*PHIw(11:15,:) )...
        -s^2 * sum( diag(Ixwg)*PHIpsi(11:15,:) );
if modesw ~= 1
  dMxwCi(1,2)=dMxwCi(1,2)+s*V*PHIthe(1,2)*(mw'*xwl*cos(labda));
end
dMxwC = dMxwC+dMxwCi;
dMxwD = sum((xwl*cos(labda)).*(Zwu.*exp(-tdelw*s)))*searw;


% dMy_wing  wing root "torsion" moment  (root is at a/c centerline),
% positive leading edge up

dMywC = (s*sum(diag(xref-(xwg+(ew-.25)*cw))*Z1w) + ...
                sum(diag(xref-(xwg+(ew-.25)*cw))*Z0w)) * theow;
dMywCi= -s^2 * sum( diag(mw.*(xref-xwg))*PHIw(11:15,:) )...
        -s^2 * sum( diag(Iywg)*PHIthe(11:15,:) );
if modesw ~= 1
  dMywCi(1,2)=dMywCi(1,2)+s*V*PHIthe(1,2)*(mw'*(xref-xwg));
end
dMywC = dMywC+dMywCi;
dMywD = sum((xref-(xwg+(ew-.25)*cw)).*(Zwu.*exp(-tdelw*s)))*searw;


% CALCULATE dMb and dMt for wing root

hulp = dMxwC*cos(labda)+dMywC*sin(labda);
dMtwC= -dMxwC*sin(labda)+dMywC*cos(labda);
dMbwC= hulp;
```

```
hulp = dMxwD*cos(labda)+dMywD*sin(labda);
dMtwD= -dMxwD*sin(labda)+dMywD*cos(labda);
dMbwD= hulp;


% TAIL SHEAR FORCE, positive downward

dZtC = (s*(Z1t+Z1d*exp(-tau*s)) + (Z0t+Z0d*exp(-tau*s))) * theot;
dZtCi= -s^2 * mt*PHIw(16,:);
if modesw ~= 1
  dZtCi(1,2)=dZtCi(1,2)+s*V*PHIthe(1,2)*mt;
end
dZtC = dZtC+dZtCi;
dZtD = Ztu.*exp(-tdelt*s)*seart + Zdu.*exp(-tau*s)*theot;


% OUTPUT MATRICES

Cy = [dnC ; dZwC ; dMbwC ; dMtwC ; dZtC];
Dy = [dnD ; dZwD ; dMbwD ; dMtwD ; dZtD];



% tffcalc.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Calculate output transfer functions w.r.t. gust input for
% a given frequency range.

clear tff

%f=logspace(-2,1.176,100)';  % from 0.01 up to 15 Hz in 100 steps logarithmically
f=[(.001:.025:3) (3:.1:15)]';  % two step widths
%f=[(.002:.05:15)]';
%f=[(.001:.005:2) (2:.05:15)]';  % two step widths


om=2*pi*f;

for k=1:length(om)

   s  = j*om(k);
   eval(theows);
   eval(theots);
   eval(GQs);
   eval(searws);
   eval(searts);
```

```
   eval(GQus);

   genout


%        We now have the system:
%        GM*x*s^2 + GK*x*s - GQ*x + GC*x = GQu*u
%        y = Cy*x + Dy*u

%        x   = [s^2*GM + s*GK - GQ + GC]^-1 * GQu*u
%        y/u = Cy*( [s^2*GM + s*GK - GQ + GC]^-1 * GQu )  + Dy

   tff(:,k)= Cy*( ((s^2*GM + s*GK - GQ + GC)\GQu) ) + Dy;

end

tff=tff.';



% tresp.m
%
% version 1.0
% 7 July 1997
%------------------------------------
%
% Calculate time response to gust inputs:
% y(t) = 1/pi * INTEGRAL( real(YS)*cos(om*t) - imag(YS)*sin(om*t) )

s=j*om;
t=[0:.02:2]';

% (1-COS) TYPE OF GUST

disp(' ');
wg = []; lg = [];
while wg == []
  wg=input('Give maximum gust speed (m/s TAS) ');
end
while lg == []
  lg=input('Give total length of gust bump (chords) ');
end
disp(' ');

tg=cw*lg/V;
ve=exp(-tg*s);
vh=(1-ve)./s;
ws=.5*wg*( vh+s.*(ve-1)./((s.^2)+(2*pi/tg)^2) );

% for Student Version:
```

```
for k=1:length(t)
  wt(k,1)=1/pi*trapz( om,(real(ws).*cos(om*t(k))-imag(ws).*sin(om*t(k))) );
end

          % the following integration is twice as fast:
%wt=1/pi*trapz( om,(diag(real(ws))*cos(om*t')-diag(imag(ws))*sin(om*t')) );
%wt=wt';

clear ys

%ys = diag(ws)*tff;

% for Student Version:
for k=1:5
 ys(:,k)= (ws).*tff(:,k);
end

%dnt=1/pi*trapz( om,diag(real(ys(:,1)))*cos(om*t')-diag(imag(ys(:,1)))*sin(om*t') )';
%dzwt=1/pi*trapz( om,diag(real(ys(:,2)))*cos(om*t')-diag(imag(ys(:,2)))*sin(om*t') )';
%dmbwt=1/pi*trapz( om,diag(real(ys(:,3)))*cos(om*t')-diag(imag(ys(:,3)))*sin(om*t') )';
%dmtwt=1/pi*trapz( om,diag(real(ys(:,4)))*cos(om*t')-diag(imag(ys(:,4)))*sin(om*t') )';
%dztt=1/pi*trapz( om,diag(real(ys(:,5)))*cos(om*t')-diag(imag(ys(:,5)))*sin(om*t') )';

% for Student Version:
for k=1:length(t)
 dnt(k,1)=1/pi*trapz( om,real(ys(:,1)).*cos(om*t(k))-imag(ys(:,1)).*sin(om*t(k)) );
 dzwt(k,1)=1/pi*trapz( om,real(ys(:,2)).*cos(om*t(k))-imag(ys(:,2)).*sin(om*t(k)) );
 dmbwt(k,1)=1/pi*trapz( om,real(ys(:,3)).*cos(om*t(k))-imag(ys(:,3)).*sin(om*t(k)) );
 dmtwt(k,1)=1/pi*trapz( om,real(ys(:,4)).*cos(om*t(k))-imag(ys(:,4)).*sin(om*t(k)) );
 dztt(k,1)=1/pi*trapz( om,real(ys(:,5)).*cos(om*t(k))-imag(ys(:,5)).*sin(om*t(k)) );
end

disp(' ');
disp('     Maximum values in time responses:');
disp(' ');
if prevm==[]
  disp(['max(|dn|)   ',num2str(sprintf('%11.5f',max(abs(dnt))))]);
  disp(['max(|Zw|)   ',num2str(sprintf('%11.4e',max(abs(dzwt))))]);
  disp(['max(|Mbw|)  ',num2str(sprintf('%11.4e',max(abs(dmbwt))))]);
  disp(['max(|Mtw|)  ',num2str(sprintf('%11.4e',max(abs(dmtwt))))]);
  disp(['max(|Zt|)   ',num2str(sprintf('%11.4e',max(abs(dztt))))]);
else
  disp('              present       previous');
  disp(['max(|dn|)   ',num2str(sprintf('%11.5f',max(abs(dnt)))),...
              '   ',num2str(sprintf('%11.5f',prevm(1)))]);
  disp(['max(|Zw|)   ',num2str(sprintf('%11.4e',max(abs(dzwt)))),...
              '   ',num2str(sprintf('%11.4e',prevm(2)))]);
  disp(['max(|Mbw|)  ',num2str(sprintf('%11.4e',max(abs(dmbwt)))),...
              '   ',num2str(sprintf('%11.4e',prevm(3)))]);
  disp(['max(|Mtw|)  ',num2str(sprintf('%11.4e',max(abs(dmtwt)))),...
```

```
            '    ',num2str(sprintf('%11.4e',prevm(4)))]);
  disp(['max(|Ztl)    ',num2str(sprintf('%11.4e',max(abs(dztt)))),...
            '    ',num2str(sprintf('%11.4e',prevm(5)))]);
end
disp(' ');
pause(1);
```

```
% pltresps.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Plot time responses of GRM model
% The function pltdash is implemented in order to make
% the printer print the dashed line actually dashed.

figure(1)
clg

orient tall
subplot(321)
plot(t,dnt)
if prevt~=[]
  hold on
  pltdash(prevt(:,1),prevt(:,2))
end
xlabel('time  [s]')
ylabel('dn  [-/-]')
title('load factor')

subplot(322)
plot(t,dzwt)
if prevt~=[]
  hold on
  pltdash(prevt(:,1),prevt(:,3))
end
xlabel('time  [s]')
ylabel('dZ_wing  [N]')
title('wing root shear force')

subplot(323)
plot(t,dmbwt)
if prevt~=[]
  hold on
  pltdash(prevt(:,1),prevt(:,4))
end
xlabel('time  [s]')
```

```
ylabel('dMb_wing  [Nm]')
title('wing bending moment')

subplot(324)
plot(t,dmtwt)
if prevt~=[]
  hold on
  pltdash(prevt(:,1),prevt(:,5))
end
xlabel('time  [s]')
ylabel('dMt_wing  [Nm]')
title('wing torsion moment')

subplot(325)
plot(t,dztt)
if prevt~=[]
  hold on
  pltdash(prevt(:,1),prevt(:,6))
end
xlabel('time  [s]')
ylabel('dZ_tail  [N]')
title('tail root shear force')

subplot(326)
axis('off')
text(0,.8,'LOADS TIME RESPONSES')
text(0,.65,'  TO (1-COS) GUST')
text(.1,.3,'____ present result','Color','y')
if prevt~=[]
text(.1,.15,'_ _ _ previous result','Color','g')
end


figure(2)
clg
subplot(121)
plot(t,wt)
if prevt~=[]
  hold on
  pltdash(prevt(:,1),prevt(:,7))
end
xlabel('time  [s]')
ylabel('w  [m/s]')
title('vertical gust speed')

subplot(122)
axis('off')
text(0,.8,'(1-COS) GUST INPUT')
text(.1,.4,'____ present input','Color','y')
if prevt~=[]
```

```
text(.1,.25,'_ _ _ previous input','Color','g')
end



% AbarGRM.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Calculates Abar, rho, N(0) and Power Spectra of the GRM model outputs

l     =762;                  % turbulence scale parameter
phiww =VKARM(f,l,V);         % two-sided (!!) von Karman spectrum

%Syy   = 2*diag(phiww)*(abs(tff).^2);     % these spectra are one-sided !!!
%Abars = sqrt( trapz(f,Syy) );

% for Student Version:
for k=1:5
 Syy(:,k) = 2*(phiww).*(abs(tff(:,k)).^2);     % these spectra are one-sided !!!
 Abars(k) = sqrt( trapz(f,Syy(:,k)) );
end

disp(' ');

if prevA==[]
  disp(['Abar_dn   ',num2str(sprintf('%11.5f',Abars(1)))]);
  disp(['Abar_Zw   ',num2str(sprintf('%11.4e',Abars(2)))]);
  disp(['Abar_Mbw  ',num2str(sprintf('%11.4e',Abars(3)))]);
  disp(['Abar_Mtw  ',num2str(sprintf('%11.4e',Abars(4)))]);
  disp(['Abar_Zt   ',num2str(sprintf('%11.4e',Abars(5)))]);
else
  disp('           present       previous');
  disp(['Abar_dn   ',num2str(sprintf('%11.5f',Abars(1))),...
              '   ',num2str(sprintf('%11.5f',prevA(1)))]);
  disp(['Abar_Zw   ',num2str(sprintf('%11.4e',Abars(2))),...
              '   ',num2str(sprintf('%11.4e',prevA(2)))]);
  disp(['Abar_Mbw  ',num2str(sprintf('%11.4e',Abars(3))),...
              '   ',num2str(sprintf('%11.4e',prevA(3)))]);
  disp(['Abar_Mtw  ',num2str(sprintf('%11.4e',Abars(4))),...
              '   ',num2str(sprintf('%11.4e',prevA(4)))]);
  disp(['Abar_Zt   ',num2str(sprintf('%11.4e',Abars(5))),...
              '   ',num2str(sprintf('%11.4e',prevA(5)))]);
end

disp(' ');

   % Calculate correlation coefficients
```

```
rhodnmbw =2*trapz(f,(real(tff(:,1)).*real(tff(:,3))+imag(tff(:,1))...
        .*imag(tff(:,3))).*phiww)/Abars(1)/Abars(3);
rhozwmbw =2*trapz(f,(real(tff(:,2)).*real(tff(:,3))+imag(tff(:,2))...
        .*imag(tff(:,3))).*phiww)/Abars(2)/Abars(3);
rhozwmtw =2*trapz(f,(real(tff(:,2)).*real(tff(:,4))+imag(tff(:,2))...
        .*imag(tff(:,4))).*phiww)/Abars(2)/Abars(4);
rhombwmtw=2*trapz(f,(real(tff(:,3)).*real(tff(:,4))+imag(tff(:,3))...
        .*imag(tff(:,4))).*phiww)/Abars(3)/Abars(4);

rhozwzt  =2*trapz(f,(real(tff(:,2)).*real(tff(:,5))+imag(tff(:,2))...
        .*imag(tff(:,5))).*phiww)/Abars(2)/Abars(5);
rhodnzt  =2*trapz(f,(real(tff(:,1)).*real(tff(:,5))+imag(tff(:,1))...
        .*imag(tff(:,5))).*phiww)/Abars(1)/Abars(5);

if prevA==[]
  disp(['rho_dnMbw ',num2str(sprintf('%11.5f',rhodnmbw))]);
  disp(['rho_ZwMtw ',num2str(sprintf('%11.5f',rhozwmtw))]);
else
  disp('              present       previous');
  disp(['rho_dnMbw ',num2str(sprintf('%11.5f',rhodnmbw)),...
              '   ',num2str(sprintf('%11.5f',prevrdnmbw))]);
  disp(['rho_ZwMtw ',num2str(sprintf('%11.5f',rhozwmtw)),...
              '   ',num2str(sprintf('%11.5f',prevrzwmtw))]);
end

disp(' ');

    % Calculate the N(0) values

Nnul(1) = sqrt( trapz(f,Syy(:,1).*(f.^2))/trapz(f,Syy(:,1)) );
Nnul(2) = sqrt( trapz(f,Syy(:,2).*(f.^2))/trapz(f,Syy(:,2)) );
Nnul(3) = sqrt( trapz(f,Syy(:,3).*(f.^2))/trapz(f,Syy(:,3)) );
Nnul(4) = sqrt( trapz(f,Syy(:,4).*(f.^2))/trapz(f,Syy(:,4)) );
Nnul(5) = sqrt( trapz(f,Syy(:,5).*(f.^2))/trapz(f,Syy(:,5)) );

disp(' ');

if prevA==[]
  disp(['N(0)_dn   ',num2str(sprintf('%7.3f',Nnul(1)))]);
  disp(['N(0)_Zw   ',num2str(sprintf('%7.3f',Nnul(2)))]);
  disp(['N(0)_Mbw  ',num2str(sprintf('%7.3f',Nnul(3)))]);
  disp(['N(0)_Mtw  ',num2str(sprintf('%7.3f',Nnul(4)))]);
  disp(['N(0)_Zt   ',num2str(sprintf('%7.3f',Nnul(5)))]);
else
  disp('              present       previous');
  disp(['N(0)_dn   ',num2str(sprintf('%7.3f',Nnul(1))),...
              '   ',num2str(sprintf('%11.3f',prevA(6)))]);
  disp(['N(0)_Zw   ',num2str(sprintf('%7.3f',Nnul(2))),...
              '   ',num2str(sprintf('%11.3f',prevA(7)))]);
  disp(['N(0)_Mbw  ',num2str(sprintf('%7.3f',Nnul(3))),...
```

```
            '    ',num2str(sprintf('%11.3f',prevA(8)))]);
  disp(['N(0)_Mtw  ',num2str(sprintf('%7.3f',Nnul(4))),...
            '    ',num2str(sprintf('%11.3f',prevA(9)))]);
  disp(['N(0)_Zt   ',num2str(sprintf('%7.3f',Nnul(5))),...
            '    ',num2str(sprintf('%11.3f',prevA(10)))]);
end

disp(' ');



% pltspecs.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Plot power spectra of GRM model

figure(1)
clg

if prevS~=[]
  nst=length(prevS);
  dash=.025;
  for k=2:6
    prevSp(:,k)=prevS(:,1).*prevS(:,k);
  end
  prevSp(:,1)=prevS(:,1);
end

orient tall
subplot(321)
semilogx(f,f.*Syy(:,1))
if prevS~=[]
  hold on
  logxdash(prevSp(:,1),prevSp(:,2))
end
axs=axis;
axis([min(f)  max(f)  axs(3) axs(4)])
fh=gca;
set(fh,'XTick',[.01 .1 1 10])
xlabel('freq  [Hz]')
ylabel('f*S_dn  [1/s]')
title('load factor')

subplot(322)
semilogx(f,f.*Syy(:,2))
if prevS~=[]
  hold on
```

```
  logxdash(prevSp(:,1),prevSp(:,3))
end
axs=axis;
axis([min(f)  max(f)  axs(3) axs(4)])
fh=gca;
set(fh,'XTick',[.01 .1 1 10])
xlabel('freq  [Hz]')
ylabel('f*S_Zw  [N^2/s]')
title('wing root shear force')

subplot(323)
semilogx(f,f.*Syy(:,3))
if prevS~=[]
  hold on
  logxdash(prevSp(:,1),prevSp(:,4))
end
axs=axis;
axis([min(f)  max(f)  axs(3) axs(4)])
fh=gca;
set(fh,'XTick',[.01 .1 1 10])
xlabel('freq  [Hz]')
ylabel('f*S_Mb  [N^2m^2/s]')
title('wing bending moment')

subplot(324)
semilogx(f,f.*Syy(:,4))
if prevS~=[]
  hold on
  logxdash(prevSp(:,1),prevSp(:,5))
end
axs=axis;
axis([min(f)  max(f)  axs(3) axs(4)])
fh=gca;
set(fh,'XTick',[.01 .1 1 10])
xlabel('freq  [Hz]')
ylabel('f*S_Mt  [N^2m^2/s]')
title('wing torsion moment')

subplot(325)
semilogx(f,f.*Syy(:,5))
if prevS~=[]
  hold on
  logxdash(prevSp(:,1),prevSp(:,6))
end
axs=axis;
axis([min(f)  max(f)  axs(3) axs(4)])
fh=gca;
set(fh,'XTick',[.01 .1 1 10])
xlabel('freq  [Hz]')
ylabel('f*S_Zt  [N^2/s]')
```

```
title('tail root shear force')

subplot(326)
axis('off')
text(0,.8,'LOADS POWER SPECTRA')
text(.1,.4,'____ present result','Color','y')
if prevS~=[]
text(.1,.25,'_ _ _ previous result','Color','g')
end
```

```
% stchresp.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Calculate time response to stochastic gust input:
% y(t) = ifft(YS)
%
% NOTE: max. number of elements in a matrix must be <=8192 for Student Version

% Transfer functions are calculated at uniformly spaced frequencies

  tg    = 34;
  nstps = 1024;
% add rv970624 for 10 sec plot
  tottime=10;
  tgeff = fix(tottime*nstps/tg);
% end add rv970624
  fst   = ((nstps/2/tg)*(0:nstps/2)/(nstps/2))';
  tst   = [0:tg/nstps:tg-tg/nstps]';

% Nyquist frequency is about 15 Hz  (nstps/2/tg)

clc
disp('Calculating transfer functions at equidistant frequencies for dn, Zw, Mbw, Mtw, Zt ...');

  omst=2*pi*fst;
  clear tffst ys
  tffst(:,1)=zeros(5,1);
  for k=2:length(omst)       % omega=0 leads to singular matrix in tff

     s  = j*omst(k);
     eval(theows);
     eval(theots);
     eval(GQs);
     eval(searws);
     eval(searts);
```

```
    eval(GQus);

    genout

    if k>1
      tffst(:,k)= Cy*( ((s^2*GM + s*GK - GQ + GC)\GQu) ) + Dy;
    else
      tffst(:,k)= zeros(5,1);
    end
  end

  tffst=tffst.';


% STOCHASTIC TYPE OF GUST, COMPLYING WITH VON KARMAN PSD

disp(' ');
sigw = []; rgen=[];
while sigw == []
  sigw=input('Give rms gust speed sigma_w (m/s TAS) ');
end
rgen=input('Put random generator seed to zero? (y/n)  [y]','s');
disp(' ');

if (rgen==[] | rgen=='y')
  rand('seed',0);
end

L  = 762;            % scale of von Karman spectrum
ws = sqrt(tg)*sigw*sqrt(VKARM(fst,L,V)).*exp(j*rand(length(fst),1)*2*pi);
% The mean of the time signal will be 0 if ws(0)=0
ws(1)=0;

for k = 1:5
  ys(:,k) = ws.*tffst(:,k);
end

% Create Hermitian sequence for fft-procedure:
ws(nstps/2+2:nstps) =conj(ws(nstps/2:-1:2));
ys(nstps/2+2:nstps,:) =conj(ys(nstps/2:-1:2,:));

% in the time domain:
wtst=2*fst(nstps/2+1)*fftshift( real(ifft(ws)) );          % 2*fst(nstps/2+1) = 1/dt

%%wtst=1/pi*trapz( omst,(diag(real(ws))*cos(omst*tst')-diag(imag(ws))*sin(omst*tst')) );
%%wtst=wtst';

disp('turbulence patch w(t) generated; calculating responses...');
disp(' ');
```

```
dntst=2*fst(nstps/2+1)*fftshift( real(ifft(ys(:,1))) );
dzwtst=2*fst(nstps/2+1)*fftshift( real(ifft(ys(:,2))) );
dmbwtst=2*fst(nstps/2+1)*fftshift( real(ifft(ys(:,3))) );
dmtwtst=2*fst(nstps/2+1)*fftshift( real(ifft(ys(:,4))) );
dzttst=2*fst(nstps/2+1)*fftshift( real(ifft(ys(:,5))) );

%%rv970624
tst=tst(1:tgeff);
wtst=wtst(1:tgeff);
dntst=dntst(1:tgeff);
dzwtst=dzwtst(1:tgeff);
dmbwtst=dmbwtst(1:tgeff);
dmtwtst=dmtwtst(1:tgeff);
dzttst=dzttst(1:tgeff);

%%rv970624 tst=tst(1:602);
%%rv970624 wtst=wtst(1:602);
%%rv970624 dntst=dntst(1:602);
%%rv970624 dzwtst=dzwtst(1:602);
%%rv970624 dmbwtst=dmbwtst(1:602);
%%rv970624 dmtwtst=dmtwtst(1:602);
%%rv970624 dzttst=dzttst(1:602);

%%dntst=1/pi*trapz(  omst,diag(real(ys(:,1)))*cos(omst*tst')-diag(imag(ys(:,1)))*sin(omst*tst')
)';
%%dzwtst=1/pi*trapz( omst,diag(real(ys(:,2)))*cos(omst*tst')-diag(imag(ys(:,2)))*sin(omst*tst')
)';
%%dmbwtst=1/pi*trapz(omst,diag(real(ys(:,3)))*cos(omst*tst')-diag(imag(ys(:,3)))*sin(omst*tst')
)';
%%dmtwtst=1/pi*trapz( omst,diag(real(ys(:,4)))*cos(omst*tst')-diag(imag(ys(:,4)))*sin(omst*tst')
)';
%%dzttst=1/pi*trapz( omst,diag(real(ys(:,5)))*cos(omst*tst')-diag(imag(ys(:,5)))*sin(omst*tst')
)';

disp(' ');
disp('     Standard deviation values in time responses:');
disp(' ');
if prevstd==[]
  disp(['std(dn)    ',num2str(sprintf('%11.5f',std(dntst)))]);
  disp(['std(Zw)    ',num2str(sprintf('%11.4e',std(dzwtst)))]);
  disp(['std(Mbw)   ',num2str(sprintf('%11.4e',std(dmbwtst)))]);
  disp(['std(Mtw)   ',num2str(sprintf('%11.4e',std(dmtwtst)))]);
  disp(['std(Zt)    ',num2str(sprintf('%11.4e',std(dzttst)))]);
else
  disp('              present        previous');
  disp(['std(dn)    ',num2str(sprintf('%11.5f',std(dntst))),...
              '    ',num2str(sprintf('%11.5f',prevstd(1)))]);
  disp(['std(Zw)    ',num2str(sprintf('%11.4e',std(dzwtst))),...
              '    ',num2str(sprintf('%11.4e',prevstd(2)))]);
  disp(['std(Mbw)   ',num2str(sprintf('%11.4e',std(dmbwtst))),...
```

```
            '    ',num2str(sprintf('%11.4e',prevstd(3)))]);
  disp(['std(Mtw)  ',num2str(sprintf('%11.4e',std(dmtwtst))),...
            '    ',num2str(sprintf('%11.4e',prevstd(4)))]);
  disp(['std(Zt)   ',num2str(sprintf('%11.4e',std(dzttst))),...
            '    ',num2str(sprintf('%11.4e',prevstd(5)))]);
end
disp(' ');
pause(1);
```

```
% pltstoch.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Plot stochastic time responses of GRM model

figure(1)
clg

orient tall
subplot(321)
plot(tst,dntst)
if prevstoch~=[]
  hold on
  pltdash(prevstoch(:,1),prevstoch(:,2))
end
xlabel('time  [s]')
ylabel('dn  [-/-]')
title('load factor')

subplot(322)
plot(tst,dzwtst)
if prevstoch~=[]
  hold on
  pltdash(prevstoch(:,1),prevstoch(:,3))
end
xlabel('time  [s]')
ylabel('dZ_wing  [N]')
title('wing root shear force')

subplot(323)
plot(tst,dmbwtst)
if prevstoch~=[]
  hold on
  pltdash(prevstoch(:,1),prevstoch(:,4))
end
xlabel('time  [s]')
```

```
ylabel('dMb_wing  [Nm]')
title('wing bending moment')

subplot(324)
plot(tst,dmtwtst)
if prevstoch~=[]
  hold on
  pltdash(prevstoch(:,1),prevstoch(:,5))
end
xlabel('time  [s]')
ylabel('dMt_wing  [Nm]')
title('wing torsion moment')

subplot(325)
plot(tst,dzttst)
if prevstoch~=[]
  hold on
  pltdash(prevstoch(:,1),prevstoch(:,6))
end
xlabel('time  [s]')
ylabel('dZ_tail  [N]')
title('tail root shear force')

subplot(326)
axis('off')
text(0,.8,'  LOADS TIME RESPONSES')
text(0,.65,'TO STOCHASTIC GUST PATCH')
text(.1,.3,'____ present result','Color','y')
if prevstoch~=[]
text(.1,.15,'_ _ _ previous result','Color','g')
end


figure(2)
clg
orient tall
subplot(221)
plot(tst,wtst)
if prevstoch~=[]
  hold on
  pltdash(prevstoch(:,1),prevstoch(:,7))
end
xlabel('time  [s]')
ylabel('w  [m/s]')
title('vertical gust speed')

subplot(222)
axis('off')
text(0,.8,'STOCHASTIC GUST INPUT')
text(.1,.4,'____ present input','Color','y')
```

```
if prevstoch~=[]
text(.1,.25,'_ _ _ previous input','Color','g')
end



% pltffs.m
%
% version 1.0
% 7 July 1997
%-------------------------------------
%
% Plot transfer functions

figure(1)
clg

pf1=[f abs(tff)];
if prevH~=[]
  pf2=[prevH(:,1)  abs(prevH(:,2:6))];
end

orient tall
subplot(321)
plot(pf1(:,1),pf1(:,2))
if prevH~=[]
  hold on
  pltdash(pf2(:,1),pf2(:,2))
end
xlabel('freq.  [Hz]')
ylabel('|H_dn|  [s/m]')
title('load factor')

subplot(322)
plot(pf1(:,1),pf1(:,3))
if prevH~=[]
  hold on
  pltdash(pf2(:,1),pf2(:,3))
end
xlabel('freq.  [Hz]')
ylabel('|H_Zw|  [N*s/m]')
title('wing root shear force')

subplot(323)
plot(pf1(:,1),pf1(:,4))
if prevH~=[]
  hold on
  pltdash(pf2(:,1),pf2(:,4))
end
xlabel('freq.  [Hz]')
```

```
ylabel('|H_Mbw|  [Nm*s/m]')
title('wing bending moment')

subplot(324)
plot(pf1(:,1),pf1(:,5))
if prevH~=[]
  hold on
  pltdash(pf2(:,1),pf2(:,5))
end
xlabel('freq.  [Hz]')
ylabel('|H_Mtw|  [Nm*s/m]')
title('wing torsion moment')

subplot(325)
plot(pf1(:,1),pf1(:,6))
if prevH~=[]
  hold on
  pltdash(pf2(:,1),pf2(:,6))
end
xlabel('freq.  [Hz]')
ylabel('|H_Zt|  [N*s/m]')
title('tail shear force')

subplot(326)
axis('off')
text(-.1,.8,'TRANSFER FUNCTION MODULI')
text(.1,.4,'____ present result','Color','y')
if prevH~=[]
text(.1,.25,'_ _ _ previous result','Color','g')
end
```

C Floppy disk with GRM m-files

The contents of the floppy disk
can be downloaded from:

http://www.nlr.nl/public/library/1998-1/97379-dcs.html