PyPk - Collection of p-k methods

Theory manual and quick reference guide

Adrien Crovato

Department of Aerospace & Mechanical Engineering

Abstract

This document provides the mathematical formulation of the main equations implemented in `PyPk` [1], version 1.1.0, December 2024. More details about the mathematical foundation can be found in the articles by Rodden [1] and van Zyl [2].

This theory manual and quick reference guide is organized as follows. Section 1 presents the mathematical formulation of the flutter problem and two variants of the p-k method implemented to solve it. Finally, section 2 gives an overview of the available API as well as its configuration parameters.

[1] https://gitlab.uliege.be/am-dept/pypk, Accessed November 2024.

Contents

1 Flutter solution methodology

1.1 Flutter equation

Neglecting internal damping, the equilibrium equations of a solid are obtained by balancing the inertial and internal forces in the solid with the external forces applied to it. The structural equations can be written as

$$\rho_{\mathrm{s}}\ddot{\mathbf{u}}_{\mathrm{s}} - \nabla \cdot \boldsymbol{\sigma}_{\mathrm{s}} = \mathbf{f}_{\mathrm{s}}, \tag{1.1}$$

where ρ_{s} is the solid density, $\boldsymbol{\sigma}_{\mathrm{s}}$ is the stress tensor, $\mathbf{f}_{\mathrm{s}}$ are the external forces and $\mathbf{u}_{\mathrm{s}}$ are the displacements. The stresses in the solid can be related to the strains and to the displacements. Equation 1.1 can then be discretized using finite elements and written in matrix form as

$$\mathbf{M}_{\mathrm{s}}\ddot{\mathbf{u}}_{\mathrm{s}} + \mathbf{K}_{\mathrm{s}}\mathbf{u}_{\mathrm{s}} - \mathbf{F}_{\mathrm{s}} = 0, \tag{1.2}$$

where $\mathbf{M}_{\mathrm{s}}$ and $\mathbf{K}_{\mathrm{s}}$ are the mass and stiffness matrices and $\mathbf{F}_{\mathrm{s}}$ is the force vector. The displacements of the solid can be expressed in the modal space by splitting them in a spatial and a time-dependent term,

$$\mathbf{u}_{\mathrm{s}} = \boldsymbol{\phi}_{\mathrm{s}}(x, y, z)\exp(\mathrm{i}\omega t), \tag{1.3}$$

where $\boldsymbol{\phi}_{\mathrm{r}}$ are the mode shapes of the solid depending solely on the space coordinates x, y, z, and ω is a frequency of vibration and t is the time. Noting that the energy related to the displacements is usually contained in the lowest frequency modes, which further allows to work with a reduced set of modal coordinates $\mathbf{q}_{\mathrm{r}}$, defined such that

$$\mathbf{q}_{\mathrm{r}} = \boldsymbol{\Phi}_{\mathrm{r}}^{\mathrm{T}}\mathbf{u}_{\mathrm{s}}, \tag{1.4}$$

where $\boldsymbol{\Phi}_{\mathrm{r}}$ is the modal matrix, containing the first mode shapes of the solid. Injecting the modal decomposition (1.4) into equation (1.2), pre-multiplying by $\boldsymbol{\Phi}_{\mathrm{r}}^{\mathrm{T}}$ and taking the Laplace transform yields

$$\omega^2\mathbf{M}_{\mathrm{r}}\mathbf{q}_{\mathrm{r}} + \mathbf{K}_{\mathrm{r}}\mathbf{q}_{\mathrm{r}} - \mathbf{F}_{\mathrm{r}} = 0, \tag{1.5}$$

where the reduced mass and stiffness matrices and the reduced force vector are given by

$$\begin{aligned} \mathbf{M}_{\mathrm{r}} &= \boldsymbol{\Phi}_{\mathrm{r}}^{\mathrm{T}}\mathbf{M}_{\mathrm{s}}\boldsymbol{\Phi}_{\mathrm{r}}, \\ \mathbf{K}_{\mathrm{r}} &= \boldsymbol{\Phi}_{\mathrm{r}}^{\mathrm{T}}\mathbf{K}_{\mathrm{s}}\boldsymbol{\Phi}_{\mathrm{r}}, \\ \mathbf{F}_{\mathrm{r}} &= \boldsymbol{\Phi}_{\mathrm{r}}^{\mathrm{T}}\mathbf{F}_{\mathrm{s}}. \end{aligned} \tag{1.6}$$

Assuming linear aerodynamics, the reduced force vector can be expressed as

$$\mathbf{F}_{\mathrm{r}} = \frac{1}{2}\rho_{\infty}U_{\infty}^2\mathbf{Q}_{\mathrm{r}}\mathbf{q}_{\mathrm{r}}, \tag{1.7}$$

where $\mathbf{Q}_\mathrm{r}$ is the generalized aerodynamic forces matrix. The flutter equation is finally obtained by injecting the expressions (1.6) and (1.7) into equation (1.5)

$$\left(\frac{U_\infty^2}{l_\mathrm{ref}^2}p^2\mathbf{M}_\mathrm{r} + \mathbf{K}_\mathrm{r} - \frac{1}{2}\rho_\infty U_\infty^2 \mathbf{Q}_\mathrm{r}(p)\mathbf{q}_\mathrm{r}\right)\mathbf{q}_\mathrm{r} = 0, \tag{1.8}$$

where l_ref is a reference length, and where $p = gk + \mathrm{i}k$, g being the damping and $k = \omega\frac{l_\mathrm{ref}}{U_\infty}$ being the reduced frequency. Since the generalized aerodynamic forces matrix depends on the reduced frequency, equation (1.8) is a nonlinear eigenvalue problem. In the present work, structural damping is approximated using the complex proportional stiffness method. The flutter equation becomes

$$\left(\frac{U_\infty^2}{l_\mathrm{ref}^2}p^2\mathbf{M}_\mathrm{r} + g_\mathrm{s}^\mathrm{i}\mathbf{K}_\mathrm{r} - \frac{1}{2}\rho_\infty U_\infty^2 \mathbf{Q}_\mathrm{r}(p)\mathbf{q}_\mathrm{r}\right)\mathbf{q}_\mathrm{r} = 0, \tag{1.9}$$

where $g_\mathrm{s}^\mathrm{i} = (1 + \mathrm{i}g_\mathrm{s})$, g_s being the estimated structural damping.

1.2 Frequency matching methods

The flutter equation (1.9) can be solved by frequency matching methods, one of the most famous being the p-k method [1]. This technique consists in iteratively solving the flutter equation by updating the reduced frequency k using the eigenvalue solution p. The algorithm of the original p-k method is given in algorithm table 1.

Algorithm 1 p-k method

for all freestream airspeed, $U_{\infty,i}$ **do**
  for all modes, j **do**
    guess the reduced frequency $k \leftarrow k_\mathrm{initial}$
    while $\Im(p) \neq k$ **do**
      compute the generalized aerodynamic forces matrix $\mathbf{Q}_\mathrm{r}(k)$
      solve the eigenvalue problem for p
      sort the eigenvalues p in ascending order of imaginary part
      keep the jth eigenvalue $p \leftarrow p_j$
      compute the new reduced frequency associated to mode j, $k \leftarrow \Im(p)$
    end while
    compute the frequency at velocity $U_{\infty,i}$ for mode j, $\omega \leftarrow \frac{U_\infty}{l_\mathrm{ref}}\Im(p)$
    compute the damping at velocity $U_{\infty,i}$ for mode j, $g \leftarrow \frac{\Re(p)}{\Im(p)}$
  end for
end for

The p-k method may experience convergence issues, as described by Jonnson [3], which may in turn impair a gradient-based optimization process. In order to mitigate these issues, a non-iterative variant of the p-k method, originally proposed by van Zyl [2] is implemented in `SDPM`. In the non-iterative version, the generalized aerodynamic forces matrix is first computed at different chosen reduced frequencies k_ref, and the eigenvalue problem is solved at each of these frequencies. Then, for each mode, the reduced frequency which is an actual solution of the eigenvalue problem is determined by linear interpolation. The algorithm of the non-iterative

p-k methods is given in algorithm table 2.

Algorithm 2 Non-iterative p-k method

choose a set of reduced frequencies k_{ref}
compute the generalized aerodynamic forces matrix $\mathbf{Q}_{\text{r}}(k)$
for all freestream airspeed, $U_{\infty,i}$ **do**
  for all frequencies, k_j **do**
    solve the eigenvalue problem for p
    sort the eigenvalues p in ascending order of imaginary part
  end for
  for all modes, l **do**
    keep the lth eigenvalue $p \leftarrow p_l$
    find the frequencies k_0 and k_1 between which $\Im(p) - k$ changes sign
    interpolate p linearly between p_0 and p_1
    compute the frequency at velocity $U_{\infty,i}$ for mode l, $\omega \leftarrow \frac{U_\infty}{l_{\text{ref}}}\Im(p)$
    compute the damping at velocity $U_{\infty,i}$ for mode l, $g \leftarrow \frac{\Re(p)}{\Im(p)}$
  end for
end for

The non-iterative p-k method requires to solve the flutter equation (1.9) for a set of reduced frequencies k_{ref}, which yields one eigenvalue solution p per mode, per reduced frequency and per airspeed. Several interpolation techniques can be used in order to find the eigenvalue that actually solves equation (1.9). In the present work, the two frequencies k_0 and k_1 between which the difference $\Delta = \Im(p) - k$ changes sign are first found. Then, the imaginary part of p is interpolated linearly as

$$\Im(p) = k_0 - \frac{k_1 - k_0}{\Delta_1 - \Delta_0}\Delta_0. \tag{1.10}$$

Similarly, the real part of p is interpolated as

$$\Re(p) = \Re(p_0) - \frac{\Re(p_1) - \Re(p_0)}{k_1 - k_0}\left(\Im(p) - k_0\right). \tag{1.11}$$

The frequency and the damping are then given by

$$f = \frac{U_\infty}{2\pi l_{\text{ref}}}\Im(p), \tag{1.12}$$

and by

$$g = \frac{\Re(p)}{\Im(p)}. \tag{1.13}$$

This yields one value for the frequency and the damping per mode and per airspeed. In the context of optimization, flutter can be prevented by requiring all the damping values to remain below a bounding curve. Following the works of Ringertz [4] and Stanford [5], the flutter constraint is formulated as

$$g \leq \begin{cases} g^\star(3U^2U^\star - 2U^3)/(U^\star)^3 & 0 \leq U < U^\star, \\ \beta(U - U^\star)^2 + g^\star & U \geq U^\star, \end{cases} \tag{1.14}$$

where $g^\star$, $U^\star$ and β are user-defined parameters. In order to efficiently drive a gradient-based optimization problem formulated using the adjoint method, the number of constraints should be limited. Therefore, the damping values are aggregated by applying a modified Kreisselmeier-Stainhauser (KS) function [6, 7] twice, firstly over the modes j, then over the airspeed i,

$$\begin{aligned} g_i &= \mathrm{KS}_{\mathrm{mode}}(g_{ij}) = \max_j g_{ij} + \frac{1}{\rho_{\mathrm{KS}}} \ln \left(\sum_j \exp \left(\rho_{\mathrm{KS}} \left(g_{ij} - \max_j g_{ij} \right) \right) \right), \\ g_{\mathrm{KS}} &= \mathrm{KS}_{\mathrm{airspeed}}(g_i) = \max_i g_i + \frac{1}{\rho_{\mathrm{KS}}} \ln \left(\sum_i \exp \left(\rho_{\mathrm{KS}} \left(g_i - \max_i g_i \right) \right) \right), \end{aligned} \tag{1.15}$$

where ρ_{KS} is a parameter controlling the aggregation: the larger the parameter, the more the function approaches the true maximum, though too large values can cause sharp changes in gradients, as noted by Jonsson et al. [3]. Note that the induced aggregation functions developed by Kennedy et al. [8] can also be used to aggregate the damping values.

1.3 Mode tracking

Since the actual solution of the flutter equation is determined by interpolating trial solutions at given reduced frequencies, it is of paramount importance that the modes are identified correctly. However, as the dynamic pressure changes, the modes may cross or coalesce. A mode tracking algorithm, initially proposed by van Zyl [9], has been implemented to alleviate this issue. The algorithm consists in comparing the eigenvectors at a given dynamic pressure to the eigenvectors obtained at the previous dynamic pressure. More specifically, a correlation matrix $\mathrm{MAC}(\mathbf{q}_{\mathrm{r}})$ is computed as

$$\mathrm{MAC}_{ij}(\mathbf{q}_{\mathrm{r}}) = \mathbf{q}_{\mathrm{r},i}^{n-1} \cdot \mathbf{q}_{\mathrm{r},j}^{n}, \tag{1.16}$$

where $\mathbf{q}_{\mathrm{r},j}^{n}$ is the new eigenvector solution of mode j and $\mathbf{q}_{\mathrm{r},i}^{n-1}$ is the old eigenvector solution of mode i. Each matrix column is then searched for the largest entry, hence determining a corresponding row. The new modes, associated to the searched columns, are then sorted against the old ones, associated to the matching rows.

1.4 Calculation of the gradients

The gradients of equations (1.9), (1.10), (1.11), (1.13) and (1.15) are required to perform gradient-based optimization. Since these equations consist of simple analytical formula, they are hand-differentiated analytically in the present work. Differentiating the generalized eigenvalue flutter problem (1.9) with respect to a parameter x yields

$$\begin{bmatrix} \frac{U_\infty^2}{l_{\mathrm{ref}}^2} 2p\mathbf{M}_{\mathrm{r}}\mathbf{q}_{\mathrm{r}} & \frac{U_\infty^2}{l_{\mathrm{ref}}^2} p^2\mathbf{M}_{\mathrm{r}} + g_{\mathrm{s}}^{\mathrm{i}}\mathbf{K}_{\mathrm{r}} - \frac{1}{2}\rho_\infty U_\infty^2 \mathbf{Q}_{\mathrm{r}} \\ 0 & \mathbf{q}_{\mathrm{r}}^{\mathrm{T}}\mathbf{W} \end{bmatrix} \begin{bmatrix} \partial_x p \\ \partial_x \mathbf{q}_{\mathrm{r}} \end{bmatrix} = \begin{bmatrix} \left(\frac{U_\infty^2}{l_{\mathrm{ref}}^2} p^2 \partial_x \mathbf{M}_{\mathrm{r}} + g_{\mathrm{s}}^{\mathrm{i}} \partial_x \mathbf{K}_{\mathrm{r}} - \frac{1}{2}\rho_\infty U_\infty^2 \partial_x \mathbf{Q}_{\mathrm{r}} \right) \mathbf{q}_{\mathrm{r}} \\ -\frac{1}{2}\mathbf{q}_{\mathrm{r}}^{\mathrm{T}} \partial_x \mathbf{W} \mathbf{q}_{\mathrm{r}} \end{bmatrix}, \tag{1.17}$$

where $\mathbf{W}$ is the matrix used to normalize the mode, which reduces to the identity matrix in the simplest case. Note that there is no dependence of $\mathbf{Q}_{\mathrm{r}}$ on p, since the eigenvalue problem is

solved using a non-iterative method. Differentiating equations (1.10) and (1.11) with respect to a parameter x yields

$$\partial_x \Im(p) = \frac{k_1 - k_0}{(\Delta_1 - \Delta_0)^2}(\partial_x \Im(p_1) - \partial_x \Im(p_0))\Delta_0 - \frac{k_1 - k_0}{\Delta_1 - \Delta_0}\partial_x \Im(p_0), \tag{1.18}$$

and

$$\partial_x \Re(p) = \partial_x \Re(p_0) + \frac{\partial_x \Re(p_1) - \partial_x \Re(p_0)}{k_1 - k_0}(\Im(p) - k_0) + \frac{\Re(p_1) - \Re(p_0)}{k_1 - k_0}\partial_x \Im(p). \tag{1.19}$$

The derivative of the damping (1.13) with respect to a parameter x is given by

$$\partial_x g = \frac{\partial_x \Re(p)\Im(p) - \Re(p)\partial_x \Im(p)}{\Im(p)^2}. \tag{1.20}$$

Finally, the derivative of the two-stage KS aggregation with respect to a parameter x is given by

$$\begin{aligned} \partial_x g_i &= \frac{\sum_j \exp\left(\rho_{\mathrm{KS}}\left(g_{ij} - \max_j g_{ij}\right)\right)\partial_x g_{ij}}{\sum_j \exp\left(\rho_{\mathrm{KS}}\left(g_{ij} - \max_j g_{ij}\right)\right)}, \\ \partial_x g_{\mathrm{KS}} &= \frac{\sum_i \exp\left(\rho_{\mathrm{KS}}\left(g_i - \max_i g_i\right)\right)\partial_x g_i}{\sum_i \exp\left(\rho_{\mathrm{KS}}\left(g_i - \max_i g_i\right)\right)}. \end{aligned} \tag{1.21}$$

Note that the parameter x typically represents the entries of the matrices $\mathbf{M}_{\mathrm{r}}$, $\mathbf{K}_{\mathrm{r}}$ and $\mathbf{Q}_{\mathrm{r}}$.

2 Quick reference guide

PyPk is configured and accessed using an Application Programming Interface (API). The complete documentation is available at https://gitlab.uliege.be/am-dept/pypk/-/wikis/home, accessed November 2024.

The list of parameters required to configure PyPk is provided below:

```
cfg = {
  'k_ref': array[float], # reference reduced frequencies
  'l_ref': float, # reference length (usually half root chord)
  'mach': float, # freestream Mach number
  'n_modes': int, # number of modes
  'g_struct': float, # structural damping (complex proportional stiffness)
  'rho_ks': float, # aggregation parameter for KS function
  'method': str, # method type ('pk' or 'nipk')
  'vrb': int # verbosity level
  'fluid': str, # fluid type ('unmatched' or 'matched_isa')
  # IF 'fluid' is 'unmatched'
  'rho_inf': float, # freesteam density
  'u_idx': array[float], # velocity index range
  'mu': float, # mass ratio
  'f_ref': float, # reference frequency
  # IF 'fluid' is 'mached_isa'
  'alt': array[float], # altitudes range
}
```

PyPk can then be initialized and used using:

```
from pypk import init_pypk
solver = init_pypk(cfg)
solver.set_matrices(m, k, q)
# Compute flutter solution
solver.compute()
solver.find_flutter()
solver.save(case_name)
solver.plot(case_name, show=True, format=fmt)
# Compute gradients (only available for NIPK)
solver.compute_gradients()
```

where m, k and q are the structural mass, structural stiffness and generalized aerodynamic force matrices. The solution and the gradients are stored as public attributes of the solver object. For example, the aggregated damping can be accessed as: solver.damp_agg.

References

[1] W. P. Rodden and E. D. Bellinger. Aerodynamic lag functions, divergence, and the british flutter method. *Journal of Aircraft*, 19(7):596–598, 1982.

[2] Louw H. van Zyl. Aeroelastic Divergence and Aerodynamic Lag roots. *Journal of Aircraft*, 38(3):586–588, 2000.

[3] E. Jonsson, C.A. Mader, J.R.R.A Martins, and G.J. Kennedy. Computational Modeling of Flutter Constraint for High-Fidelity Aerostructural Optimization. In *AIAA SciTech Forum*, San Diego, CA, USA, 2019. AIAA.

[4] U. Ringertz. On structural optimization with aeroelastic constraints. *Structural optimization*, 8(1):16–23, 1994.

[5] B.K. Stanford. Role of Unsteady Aerodynamics During Aeroelastic Optimization. *AIAA Journal*, 53(12):3826–3831, 9 2015.

[6] G. Kreisselmeier and R. Steinhauser. Systematische Auslegung von Reglern durch Optimierung eines vektoriellen Gütekriteriums. *Automatisierungstechnik*, 27(1-12):76–79, 1979.

[7] N.M.K. Poon and J.R.R.A. Martins. An Adaptive Approach to Constraint Aggregation Using Adjoint Sensitivity Analysis. *Structural and Multidisciplinary Optimization*, 34(1):61–73, 2007.

[8] Graeme J. Kennedy and Jason E. Hicken. Improved constraint-aggregation methods. *Computer Methods in Applied Mechanics and Engineering*, 289:332–354, 2015.

[9] Louw H. van Zyl. Use of eigenvectors in the solution of the flutter equation. *Journal of Aircraft*, 30(4):553–554, 1993.