1. Clear memory after each run

Agreed, this should be done. Can also add in options to reduce the level of detail kept for runs in memory (i.e. no need to keep all results onces stats computed perhaps).

1. Group all manual inputs at top of script, e.g. train and speed selection

My update addresses this already. I'm also seeking to use parallel processing (i.e. utilise all 4 cores on our laptops) which should improve runtimes. But I can't get this to work well from Spyder currently…

1. Automatically calculate length of bridge

My update will address this.

1. Automatically count number of output positions (rather than editing *responses2Plot=[0,1,2,3]*)

My update addresses this already, using optional 'None' input to plot all defined responses.

1. Start with check for incomplete or discontinuous input files

My update makes more checks so should catch errors early

1. During solving, check for 'NaN' results and terminate loop if present

The main update I've made to the tstep solver class is to use Scipy's Runge-Kutter solver, rather than manually code-in the linear acceleration method (as I had done previously). So numerical stability should be less of an issue – and issues will be reported by the Scipy solver.

1. Easy way to export results (peak accelerations or displacement/acceleration time histories) to Excel

I'll resist linking to Excel, but can add a function to output these to a csv file with headers

1. Graphing functions, e.g. quick way to plot force function and time histories (with titles, formatting etc.). Ability to specify which modes/trains/speeds to include

My update gives improved graphing. We can discuss specifics of what you require perhaps. The way I've coded it means that plots can be made to subplots within a custom figure, for example.

1. Add 'frequency limit'. e.g. Eurocode says only modes up to 30Hz need to be included – option to automatically exclude all modes >30Hz?

This can be done quite easily. I'll include as an option in my update.

1. Facility to add limit lines to graphs (e.g. 5m/s2 Eurocode acceleration limit)

I'll include this functionality via an optional input argument to the plotting routine. Note that my routine knows nothing about the units /

1. Default t\_epilogue = 1 sec

This is handled via an optional argument: these can always be overridden from outside. Note the philosophy of Python is "batteries included" i.e. where possible provide optional arguments with default values to make the function work!

More generally I will include an Excel file in the network distribution folder for my "dynsys" package, to allow requests like this to be logged and closed-out over time. This provides a good "starter for 10"!