gropt Documentation

Mike Loecher

Apr 09, 2023

Contents:

CHAPTER 1

Overview

GrOpt is a toolbox for MRI Gradient Optimization (GrOpT). Rather than analytical combinations of trapezoids, the software uses optimization methods to generate arbitrary waveforms that are (hopefully) the most time optimal solution.

The main workhorse of the package are the optimization routines written in C, which for final applications are designed to be dropped into a pulse sequence as needed. Additionally, wrappers for Python and Matlab are provided, as well as some demo cases to show how the software can be used, and to prototype sequences and combinations of constraints.

1.1 Installation

GrOpt is written in C, with many wrapper and helper functions written in Matlab and Python.

1.2 C Compilation

The main optimization routines are written and can be directly compiled. Example usages are shown at the bottom of `src/optimize_kernel.c`. All of the functions that are would be called by the user are found in `src/optimize_kernel.c`

1.3 Python

The optimization routines have been wrapped with Cython, they can be built simply running in the python/ directory:

```
python setup.py build_ext --inplace
```

Which will build the python module named `gropt` in the python directory, which can then be imported into any python code.

Example jupyter notebooks are provided in the python/ directory that show many ways how the module can be used.

1.4 Matlab

There is set of mex functions to use for calling the optimization routines from Matlab. They can be compiled by running the file `make.m` from within the matlab directory.

The matlab/ directory then has sample scripts that show the verious ways to call the optimization functionss.

1.5 Constraints

Below are a list of constraints and documentation about their usage and input formats.

Where possible, the units are all Tesla, meters, and seconds (though occasionally ms will come up for helper functions where it might make more sense)

Most constraints are subject to a 1% cushion to help convergence of the optimization, so many constriants might be 99% of their actual value.

1.5.1 Gradient Strength

Maximum gradient strength (G_{max}) is defined in $\frac{T}{s}$

This constraint simply clips the gradient waveforms so that all values must be in the range $(-G_{max}, G_{max})$

1.5.2 Slew Rate

Maximum slew rate (SR_{max}) is defined in $\frac{T}{(m \cdot s)}$

This constraint constrains $\frac{dG}{dt}$ so that all values must be in the range $(-SR_{max}, SR_{max})$

1.5.3 Moments

Moments are controlled with an array of doubles of length N_momentsx7, where N_moments is the number of moment constraints to be applied. Any number of constraints (N_moments) can be added in this array.

The 7 options in a constraint are (sequentially):

- **Axis of constraint** 0, 1, or 2 based on which axis to apply the constraint on, when more than one axis of gradients are being computed.

  Not currently implemented

- **Moment order** What order of moment the constraint is. i.e. 0 = M_0, 1 = M_1, 2 = M_2.

  Technically any number should work, though nothing higher than 2 is routinely tested.

- **t=0 offset** This field gives the offset in **seconds** between the first point in the waveform, and the actual t=0 time for the moment integrals.

  e.g. If there is a 2 ms rf pulse before the waveform that is not included in the waveform, but you would still like to calculate moments from the excitation, this argument could = 1e-3 (1 ms)

- **Start time** Defines the beginning of the range of the waveform that the moment should be constrained over

  -1 means the very beginning of the waveform (default behavior, to calculate for the entire waveform)

  (0, 1) gives a time in **seconds** relative to the start of the waveform (not the previous argument reference)

  >1 is cast to an integer and gives the index of the point to start the calculation

  Not currently implemented

- **End time** Defines the end of the range of the waveform that the moment should be constrained over

  -1 means the very end of the waveform (default behavior, to calculate for the entire waveform with -1 above)

  (0, 1) gives a time in **seconds** relative to the start of the waveform (not the previous argument reference)

  >1 is cast to an integer and gives the index of the point to end the calculation

  Not currently implemented

- **Desired Moment** The desired moment, with units $\frac{mT \cdot ms^{N+1}}{m}$, where N is the moment order (see second argument)

- **Moment Tolerance** How far off the waveform can be from the desired moment, in the same units.

  The optimization will very frequently use exactly all of this tolerance

1.5.4 Eddy Currents

Eddy current constraints are controlled with an array of doubles of length N_eddyx4, where N_eddy is the number of eddy current constraints to be applied. Any number of constraints (N_eddy) can be added in this array.

The 4 options in a constraint are (sequentially):

- **Time constant of constraint** The time constant in milliseconds of the constraint to be applied
- **Target eddy current magnitude** The target value of the constraint (0 for nulling, but can be set to other values). the units here are basically arbitrary.
- **Tolerance** The tolerance in the target value to be used. The units are somewhat arbitrary, but you can start around 1.0e-4
- **Mode** 0.0 to constrain the instantaneous eddy currents at the end of the waveform. 1.0 to constrain the sum of eddy current fields across the whole waveform.

1.6 Usage

Usage notes for the C, Python, and Matlab tools

1.6.1 C Usage

The two primary calls to the C library are the run_kernel_diff() and minTE_diff() functions.

Both take basically the same arguments, except run_kernel_diff() takes a fixed TE, and minTE_diff() takes a b-value to search for.

For usage examples, see the end of src/optimize_kernel.c, basic usages of both functions are shown.

Arguments

Note that this is just a list of all possible arguments, the order might be slightly different for a given function, see definitions in src/optimize_kernel.c

double **G_out Pointer to the array where the output will be stored. This will be allocated within the function.

int *N_out Pointer to an integer specifying the size of G_out after completion of the optimization.

double **ddebug Pointer to the array where debug output information will be stored. It will have a fixed length 100 (but you should probably check that to see if it has changed)

int verbose 0 for no message, >0 for debug messages to console

int N The size of the output gradient array

double dt The raster time of the gradient waveform in seconds, i.e. the distance between timepoints.

double gmax The maximum gradient amplitude in T/s.

double smax The maximum slew rate in T/m/s.

double TE The TE of the waveform in MILLIseconds. For diffusion waveforms, this takes into account T_readout. For non-diffusion it is just the length of the waveform, i.e. N*dt.

int N_moments The number of moment constraints that will be used.

double *moments_params An array describing the moment constraints, with 7 values per constraint as described *here*. So it is an array of N_moments*7 doubles.

double PNS_thresh PNS threshold with the single exponential model, any decimal should work. To disable the constraint use -1.0 (or any negative number).

double T_readout, double T_90, double T_180 Timings of the readout prewinder, initial excitation pulse, and 180 pulse, in MILLIseconds.

double bval_weight, double slew_weight, double moments_weight Initial weights for the various constraints, these can be kept at defaults, or all at 1.0, and everything will work well.

double bval_reduce How much to change the above weights when convergence has slowed down too much. Dafault to around 10.

double dt_out Adds a final linear interpolation to the waveform to reach dt_out in seconds.

int N_eddy Number of eddy current constraints, can be 0 to disable

double *eddy_params An array describing the eddy current constraints, with 4 values per constraint as described *here*. So it is an array of N_moments*4 doubles.

int is_Gin 1 if there is an input gradient waveform for warm starting the optimization, 0 to disable (default).

double *G_in An array holding initialization values for the G vector in the optimization

double search_bval When used in run_kernel, this gives a bvalue maximum, so iterations with a higher b-value are immediately stopped. It can be set to -1 to disable.

When referred to in minTE_diff(), it is the bvalue we are searching for.

int N_gfix Number of fixed values in G. 0 is off (which technically enforces G=0 at the beginning and end), 2 informs the function that *gfix only has a start and end G value. N_gfix = N says that *gfix is a full array.

double *gfix G values to set as fixed. Can be 2 entries for start and end. If full array, then big negative numbers mean NOT fixed, so fill array with -99999 and then fill in fixed values.

double slew_reg This describes the slew rate minimization that is applied. The real amount is actually a multiplier of (1-slew_reg), so 1.0 is nothing, and 0.0 is complete regularization. Default is 1.0 right now but it might switch to 0.99 for smoother waveforms.

1.7 Citations

Citations for our work here