RB-SFA: High Harmonic Generation in the Strong Field Approximation via *Mathematica*

Usage and Examples

Loading the package

You can use this software

- within the RB-SFA notebook itself by simply running the initialization cells of that notebook, or
- from an external notebook by loading it as a package.

In the latter case, place a copy of the package file RB-SFA.m on the same directory as your notebook and run the loading command

```
Needs["RBSFA`", FileNameJoin[{NotebookDirectory[], "RB-SFA.m"}]]
```

You can also call the package from another directory by suitably modifying the directory call. If you plan on using this package in the long term you can use the File > Install prompt, in which case the package is simply loaded as Needs["RBSFA`"], though this is not particularly recommended. (A better choice is to include a soft link called RBSFA.m in your $UserBaseDirectory/Applications/ directory to the file RB-SFA.m. This works just fine and is easy to undo if required.)

To print the version of the package in use, use the command

```
$RBSFAversion
```

```
RB-SFA v2.0.7, Thu 28 Apr 2016 18:26:31
```

```
$RBSFAcommit
```

```
commit 4911698072133031a5c18c7be90179d572b9bc16
Author: Emilio Pisanty <pisanty@mbi-berlin.de>
Date:   Thu Apr 28 17:54:37 2016 +0200
    Overhauled RBSFAversion to $RBSFAversion

    Usage RBSFAversion[] is now deprecated.
```

```
Quit
```

```
Names["RBSFA`Private`$*"]
```

```
{RBSFA`Private`$RBSFAdirectory}
```

```
$RBSFAdirectory
```

```
/home/episanty/Work/CQD/Project/Code/RB-SFA/
```

```
StringJoin[Riffle[ReadList["!cd " <>
    "/home/episanty/Work/CQD/Project/Papers/Contour freedom and low energy structures" <>
    " && git log -1", String], {"\n"}]]


"/home/episanty/Work/CQD/Project/Papers/Contour freedom and low energy structures"

StringJoin[
 Riffle[ReadList["!cd " <> "/home/episanty/Work/CQD/Project/Papers/Contour\ freedom\ and\
      low\ energy structures/slalomincomplextime" <> " && git log -1", String], {"\n"}]]


StringJoin[
 Riffle[ReadList["!cd " <> "/home/episanty/Work/CQD/Project/Papers/Contour\\ freedom\\ and\\
      low\\ energy\\ structures/Physics\\ paper/" <> " && git log -1", String], {"\n"}]]
commit 34438868ee04dc09b99c73fc5d5c66c61022c4da
Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>
Date:   Mon Feb 8 22:57:57 2016 +0000
    Put W Becker back in acknowledgements.

FileNameJoin

"this \\abc\\def\\ghi\\nop /home/episanty/Work/CQD/Project/Papers/Contour
  freedom and low energy structures/Physics paper is a file name"



DirectoryName[RBSFA`Private`$RBSFAdirectory]
/home/episanty/Work/CQD/Project/Code/

Run["cd /home/episanty/Work/CQD/Project/Code/RB-SFA/ && git log-1"]
256

Read["!cd /home/episanty/Work/CQD/Project/Code/RB-SFA/ && git log-1"]
EndOfFile

ReadList[
 "! cd /home/episanty/Work/CQD/Project/Code/RB-SFA/ && touch file  && git log -1", Record]
{commit 4911698072133031a5c18c7be90179d572b9bc16,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>,
 Date:   Thu Apr 28 17:54:37 2016 +0200,     Overhauled RBSFAversion to $RBSFAversion,
     ,     Usage RBSFAversion[] is now deprecated.}

ReadList["! echo \"hello\" && cd /home/episanty/Work/CQD/Project/Code/RB-SFA/
   && touch file  && git log && echo \"bye\"", Record]
{hello, commit 4911698072133031a5c18c7be90179d572b9bc16,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>,
 Date:   Thu Apr 28 17:54:37 2016 +0200,     Overhauled RBSFAversion to $RBSFAversion,
     ,    Usage RBSFAversion[] is now deprecated.,
 commit 99cf2220602588790f7c77cb0ee4efe6daab5129,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Wed Mar 30 17:22:38 2016 +0200,
     Updated version number., commit 9c5d8db58631670ee856de6b34ed9747dd3caa35,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Wed Mar 30 17:17:06 2016 +0200,
     Updated pdfs., commit ef209191989490718d339e5564d73e6bfcaae0e6,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>,
 Date:   Wed Mar 30 17:01:12 2016 +0200,    Revert "Stabilized RunInParallel",
     ,    This reverts commit 264f7d49989c515486f431488f5f552292a0faa7.,
 commit 3374b07818eb98faafb32d594b13a53ede66a99f,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Wed Mar 30 17:00:11 2016 +0200,
     Stabilized RunInParallel,    ,    Discarded old RunInParallel options and
   replaced them with a catchall version. Documented the inactive parallelization.,
 commit c16b12ba2f607e1bdb500a97ff547459f77d1c73,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Thu Mar 24 18:25:13 2016 +0100,
     Commit to test upload, commit ac4a86e28ee8f386eea775ac0b6351228fe9f29f,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Thu Mar 24 17:19:57 2016 +0100,
     Getting away from the craziness, commit cde8f572d1e0734704b6e19c8c683f1b984ca57a,
 Merge: 58db418 8703f25, Author: Emilio Pisanty <pisanty@mbi-berlin.de>,
 Date:   Thu Mar 24 17:19:07 2016 +0100,    Merge branch 'InactiveParallelization1',
     ,    Conflicts:,        RB-SFA.m, commit 58db4189d852a0af9a52f126831a532933def3a6,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Thu Mar 24 16:05:16 2016 +0100,
     Added inactive version, abstracted TableCommand.,
 commit 8703f2576638f72ad4705b275a91a5bd796009d3,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Wed Mar 23 19:45:01 2016 +0100,
     Fixed inactive parallelization., commit 81bec55cab42fcb4d894be7709d95d83c4d9b95c,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Wed Mar 23 19:37:39 2016 +0100,
     Added inactive parallelization., commit 2e5fcbf07719cc9039a9aa64a51f797e78bba095,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Wed Mar 23 15:58:41 2016 +0100,
     Testing parallelization options., commit ffd0358a194c4a53f0dbb16ec7e1344026e89019,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Wed Mar 23 13:50:27 2016 +0100,
     Changed dipole[t] -> {t, dipole[t]} in loop.,
 commit 488c742e0722f13849ad1a284dc03a4bd957ad50,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Wed Mar 23 13:47:56 2016 +0100,
     Testing the upload., commit b4d008c61115af0a8876271eb8aa6d9815df7d37,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Mon Mar 21 16:58:02 2016 +0100,
     Added single-run parallelization., commit f9a354fff5be8b0f80b350109619e4e7e0096b8b,
 Author: Emilio Pisanty <pisanty@mbi-berlin.de>, Date:   Mon Mar 21 15:00:41 2016 +0100,
     Added $DistributedContexts redefinition,    ,
     Added context to $DistributedContexts list, so it is no longer necessary,
     to DistributeDefinitions[] before running parallel computations.,
 commit 2bb2135f8f1559692404c48d869a8c1c1191136b,
 Author: Emilio Pisanty <episanty@gmail.com>, Date:   Tue Mar 15 20:01:52 2016 +0100,
     Hopefully an admin commit, commit 1c61e9df0333687ddc736fbc5308e3551b3266ba,
 Author: Emilio Pisanty <episanty@gmail.com>, Date:   Thu Mar 3 14:53:11 2016 +0100,
     Documented version command. Update to v2.0.2 to,
     cap recent changes., commit d54bb6b50236550733ec3dfcb30de1e437788fb9,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Feb 18 23:30:58 2016 +0000,    Trivial edit to test laptop.,
 commit ceef7a72837de71ed0aa2748ebf77f998833e0ed,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 17 13:19:59 2016 +0000,
     Added option to use separate ionization and recombination dipoles.,     ,
     Made the conjugation of the recombination dipole explicit to keep the t',
     integrand analytic., commit 3927fc4a1518a30be469bcda3bca91db53f2b7d2,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 10 16:08:45 2016 +0000,
     Changed Norm[p]^2 to Total[p^2] in DTMEs to keep integrand analytical.,
 commit 93001c7b59e691c819cd9fe08740396b9c3e834d,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 10 13:30:19 2016 +0000,
     Changed Norm to Total[#^2] in action to keep it analytical.,
 commit 04040055b1c89edbcddb360afa28113950df8fc0,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Feb 5 17:28:43 2016 +0000,
     Added benchmarks for decoupled IntegrationPointsPerCycle to docs.,
 commit 1b1cda089fa44d48b091cae29d0f94b251c8effe,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Feb 5 16:58:20 2016 +0000,
     Cleaned documentation of numerical nondipole preintegrals.,
 commit 0bfd9b23db69f53c37f7346a1518ce487b2b7b2f,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Feb 5 15:41:39 2016 +0000,     Fixed PDE-based nondipole preintegrator.,
 commit 3720487a4a2366dc932f801db6b936c56d491af4,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Feb 4 23:47:11 2016 +0000,
     Incorporation and initial tests of IntegrationPointsPerCycle.,
 commit 287f5a1ed5135ed7b526dff0ace7607bcaba4363,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Feb 4 22:51:55 2016 +0000,
     Removed obsolete gridPointQ from the code. Cleaned up code & docs.,
 commit 62a522b62c8afed9bfe5ce7febf9b028be351189,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Feb 4 22:40:22 2016 +0000,
     Cleaned up numerical preintegrals documentation.,
 commit 1d3cffac11aa867cce5f70f01e523d9cc83ca4ea,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Feb 4 22:36:15 2016 +0000,
     Test of new preintegrator. Still not reliable, but currently takes 35s on the,
     test (instead of 7min, up from 15s on the dipole case), and this is expected,
     to remain OK if the PDE domain gets fixed.,
 commit c12a541036ff75f25ddf1f492bf4cacbcb9060d8,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Feb 4 22:33:11 2016 +0000,
     Cleaner implementation of PDE-based preintegrator, moved from,
      "NewNumeric" to "Numeric", deleted old "Numeric" option.,
 commit 7fdf05f288f03db049a88a86e4b4fb334944343b,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Feb 4 22:27:23 2016 +0000,
     Messy mid-way for PDE-based nondipole preintegrals.,
 commit 8ff4c684872fac02d52e01ae1d3ce202c3db1b52,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 3 22:18:28 2016 +0000,
     Successful test of the new preintegrals - but very slow.,
 commit 8e2cae45159e2159a6d06d4162e07664a89597cc,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 3 22:06:06 2016 +0000,
     Diagnostics on the integration errors - they appear harmless.,     ,
     They happen at taupre close to tInit when t and tt are nowhere near. I'm,
     cutting down the (overbroad) integration range for that case.,
 commit 6ddef94881322a371e14e2dbba36678cdb5689e1,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 3 21:51:00 2016 +0000,     Found some integration errors, investigating.,
 commit 028bf32dc9a7d309bb7667731a56f4da5a57ea9c,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 3 21:35:04 2016 +0000,     Fixed minor problem on variable collision.,
 commit 9e670550a5282a7bc2d298e12ed573683b50e35b,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 3 21:21:08 2016 +0000,
     Tentatively working NDSolve-based numerical preintegration.,
 commit d5a4ca70338fd88c5316e947706d4598af858ac4,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 3 18:26:49 2016 +0000,
     Attempt at ParametricNDSolve in numerical preintegrator.,
 commit 8785d3f980d103599cf423d54a40a390955e7ed0,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 3 15:41:16 2016 +0000,
     Made preintegrator more flexible on the dimensions of the default zero,
     matrix - towards a better numerical preintegrator.,
 commit 59fe4c701e68fa4214e434a975d8dd3e9fca7af5,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Feb 3 14:48:22 2016 +0000,     Minor addition to getIonizationPotential.,
 commit ef993bb4b6007ca0fbd746b6bdeff0bfb2e3ad0e,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Dec 11 15:03:04 2015 +0000,
     Fixes for the getIonizationPotential implementation.,
 commit 976ea2918f3189939a690c5ca7522b94caa22cb1,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Dec 2 21:34:50 2015 +0000,
     Added section on long-trajectory gating to Usage and Examples file.,
 commit 18d2876343c0be3d4a0fb5940b73c33c0fd78207,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Dec 2 17:30:57 2015 +0000,
     Made getIonizationPotential a standalone function. Removed residual,
     nGateRamp mention in a Protect statement.,
 commit 441d065c0a9b822964feb4984023ace4feec4a92,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Tue Oct 27 15:46:24 2015 +0000,
     Added PointNumberCorrection option to timeAxis.,
 commit 5526f0d427fcec628e698b2cded59412f8f898c6,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Oct 16 16:53:16 2015 +0100,
     Regressed numerical preintegrals for nondipole case.,
 commit cc8589131f245b0b03e88f791f86455af9cea094,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Oct 16 16:31:36 2015 +0100,
     Tentative updating of numerical preintegrals to reflect change in spec.,
 commit a59ad16e3621209f8828ea59f75d54e15abe39d8,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Oct 16 16:31:05 2015 +0100,
     Minor change to timestamp saver to prevent it mucking with the selection.,
 commit 41baa74e16ea569fc4c4493a048df128d80687aa,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Oct 16 15:53:00 2015 +0100,
     Fixed nondipole periodicity issue. Benchmark in Usage file shows the fix. Added
   version-printing command., commit 3731094cf8b1834173d33e9b8a2cfa2ae2ab0ec6,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Oct 16 11:55:52 2015 +0100,
     Updated spec to reflect fixes for nondipole periodicity issue.,
 commit 8cdc3e6045a52155b5822ae61485076309faa343,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Oct 16 11:34:17 2015 +0100,
     Added Nondipole Periodicity section showing the problem.,
 commit 245cf3e2e7baf75dd82c855d15c163b4c0cd0962,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Wed Oct 7 16:14:03 2015 +0100,     Added A[t] and GA[t] to Verbose->2 call.,
 commit e811e8c5562f8897be858a730245baf4fa36508a,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Mon Oct 5 20:18:58 2015 +0100,     Minor readme modifications,
 commit 671b0b53f97e11b5abb6f5bf8ded382915150f50,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Oct 2 18:19:40 2015 +0100,     Added timings and memory usage section.,
 commit 4331a5b147828fa4aa0d9eed710cf52a8ca75d93,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Fri Oct 2 13:01:16 2015 +0100,     Modified .gitignore,
 commit 87da1afa9792c47e4dd2d5d221ff945eccc57368,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Oct 1 19:25:29 2015 +0100,
     Added benchmarking against Kylstra et al. (2001).,
 commit 336d8365f68ef07b3a845ecc104db4983f7f2350,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Thu Oct 1 17:36:27 2015 +0100,     Added Target option. Added gaussian DTME.
   Split into package + Usage and Examples. Added crossed-beams example as benchmarking.,
 commit b6fba2c3f897b6a28ada6352c3ea0f5dafcfcf11,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Tue Sep 29 18:15:18 2015 +0100,
     Expanded handling of dipole transition matrix elements.,
 commit 1a72a71b926781bc3a4d4450b252b41dba93632b,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Tue Sep 29 17:33:38 2015 +0100,     Cleaned up options npp, num and omega,
 commit 037634e5078c9bd98bc7d3e2bb1523e631d3cea5,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Mon Sep 28 17:40:58 2015 +0100,     Minor corrections to text,
 commit 3ee54511b12a40fbaeefc2d06964c69a21634781,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Mon Sep 28 17:35:15 2015 +0100,     Update README.md,
 commit d684ee96486b750418bff85ab5a5308e88246e32,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Mon Sep 28 17:34:57 2015 +0100,     Update README.md,
 commit 79738fbda2d571afdfb1203a391860da9430be33, Merge: 8d5cf5c ad18d59,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Mon Sep 28 17:32:34 2015 +0100,
     Merge branch 'master' of github.com:episanty/RB-SFA,     ,     Conflicts:,
         README.md, commit 8d5cf5c23ad63d3b51d6b7df5ece4b58b37bea49,
 Author: Emilio Pisanty <e.pisanty11@imperial.ac.uk>,
 Date:   Mon Sep 28 17:29:59 2015 +0100,     Version 2.0.,
 commit ad18d59c23a71f1e93fee2620d91ef0a283a66e9,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Fri Aug 22 20:17:33 2014 +0100,     Update README.md,
 commit 6f0d167f08fcbd213a1e02b00b7c42dcb0374948,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Fri Aug 22 20:17:20 2014 +0100,     Update README.md,
 commit 180c6293ba8eb629d20134046ac7e2b8027558e1,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Fri Aug 8 12:51:33 2014 +0100,     Update README.md,
 commit 54a32ce057a4d8fe951fcc60f33fd24c213f6311,
 Author: episanty <episanty@gmail.com>, Date:   Thu Aug 7 19:02:18 2014 +0100,
     Minor changes to readme. Mostly testing the remote push.,
 commit 8622608d1d5248cc62eaefef02cd4c71bd74ab8d,
 Author: episanty <episanty@gmail.com>, Date:   Thu Aug 7 18:53:09 2014 +0100,
     Small changes to readme., commit d6052b9ac28b5c4bc6685b228c058f28a7d08349,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Thu Aug 7 16:22:49 2014 +0100,     Update README.md,
 commit dac5fdda8c74b08f937eeabacd5f41f0246789cd,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Thu Aug 7 16:20:45 2014 +0100,     Cleaned up README,
 commit 47211cc940157f92e474ffb7e3d23e629acd0709, Merge: 5e0ee3e 8509b3c,
 Author: episanty <episanty@gmail.com>, Date:   Thu Aug 7 16:18:21 2014 +0100,
     Merge branch 'master' of https://github.com/episanty/RB-SFA,     ,
     Conflicts:,         README.md, commit 5e0ee3e7c0dd016b9ac1edbda932c2d05587d0f9,
 Author: episanty <episanty@gmail.com>, Date:   Thu Aug 7 16:14:10 2014 +0100,
     Added .gitignore file., commit bf1a2ee90bf52a0db65cb6e4e6422f5efa69f982,
 Author: episanty <episanty@gmail.com>, Date:   Thu Aug 7 16:03:55 2014 +0100,
     Initial upload, commit 8509b3c7e23840a8dc2e81a132a1d103a6e39d4c,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Thu Aug 7 14:19:04 2014 +0100,     Update README.md,
 commit ebf062de427bbf1e5279c1452ec63a204946ca5b,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Thu Aug 7 14:17:25 2014 +0100,     Update README.md,
 commit 4b3d2f03aac18e240ab57f099727b2b333c04199,
 Author: E Pisanty <episanty@users.noreply.github.com>,
 Date:   Thu Aug 7 13:37:34 2014 +0100,     Initial commit, bye}
```

```
Import["!cd /home/episanty/Work/CQD/Project/Code/RB-SFA/ && echo \"hello\"", "Text"]
```

```
hello
```

Simple usage

For basic usage, simply call the main numerical integrator, makeDipoleList, with the vector potential you want to use, and provide any parameters you wish to specify using the FieldParameters option.

```
AbsoluteTiming[
 simpleDipole = makeDipoleList[
    VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}], FieldParameters → {F → 0.05, ω → 0.057}];
]
```

```
{3.01082, Null}
```

Calling the function with insufficient parameters will produce error messages:

```
makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}]]
```

```
makeDipoleList::pot :
  The vector potential A provided as VectorPotential→Function[t, {F Sin[ω t]/ω, 0, 0}] is incorrect or is missing FieldParameters.
      Its usage as A[4.710127010714186`] returns {F Sin[4.71013 ω]/ω, 0, 0} and should return a list of numbers.
```

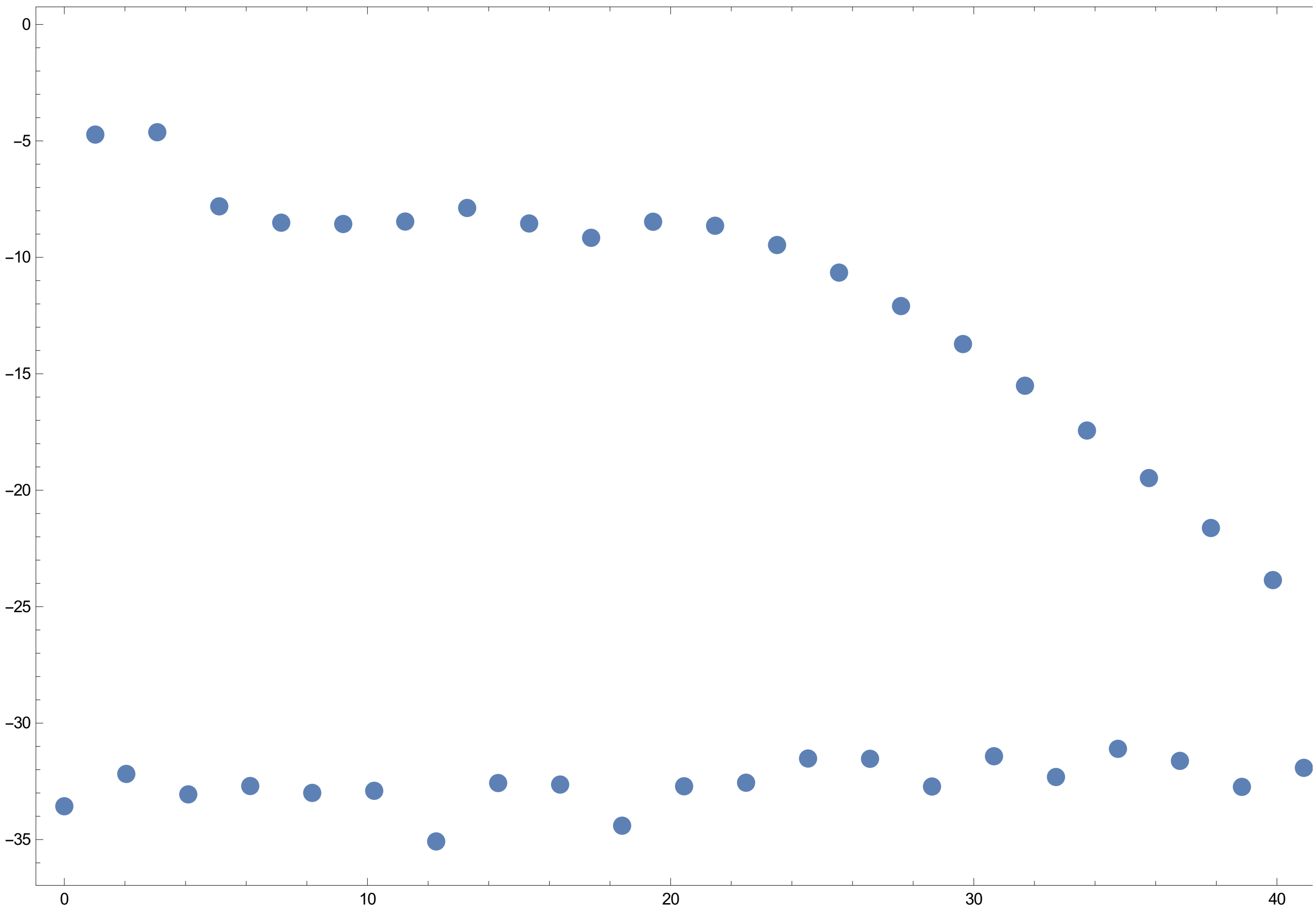
```
$Aborted
```

The symbol ω is taken to be the carrier frequency, and is set by default to $\omega = 0.057$ atomic units, corresponding to a wavelength of 800 nm. If the carrier frequency is changed, this must be specified on **both** the field parameters and the explicit option for the integrator, as

```
makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
 FieldParameters → {F → 0.05, ω → 0.0456}, CarrierFrequency → 0.0456]
```

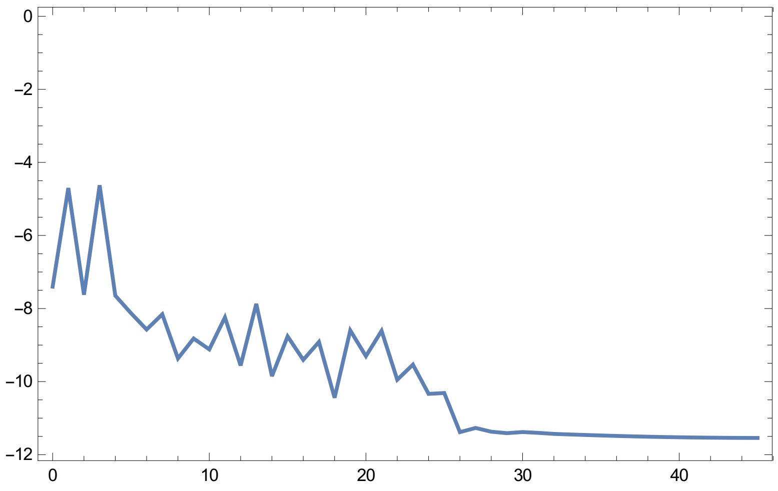
To see the spectrum, use the getSpectrum and the spectrumPlotter commands, such as

```
spectrumPlotter[getSpectrum[Most[simpleDipole]], Joined → False]
```



Note here the use of Most on the dipole when a monochromatic field is indicated. This ensures that the signal is actually periodic (i.e. it eliminates repetition between the initial and final points, which are separated by exactly one period). If this is not done, the spectrum is much noisier:

```
spectrumPlotter[getSpectrum[simpleDipole], ImageSize → 400]
```



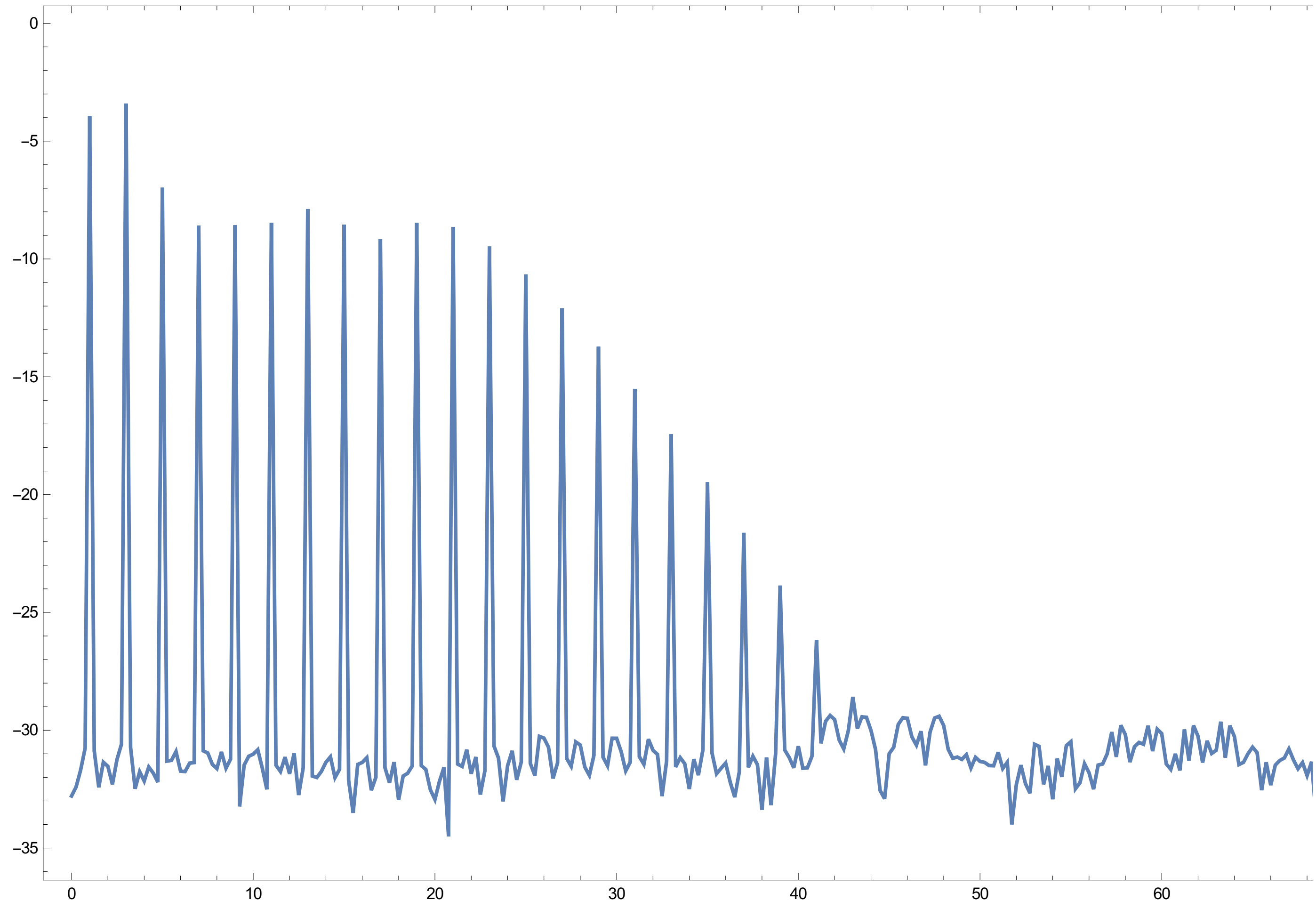
The default options are built for a periodic pulse for which simple functions of the vector potential can be integrated analytically, and for which only a single period of integration is necessary. More periods can be specified using the TotalCycles option. Similarly, the PointsPerCycle option controls the number of points per period.

```
AbsoluteTiming[
 biggerDipole = makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
    FieldParameters → {F → 0.05, ω → 0.057}, TotalCycles → 4, PointsPerCycle → 150];
]
```

```
{27.772, Null}
```

To get a correct spectrum plot, give these settings to the spectrum plotter.

```
spectrumPlotter[getSpectrum[Most[biggerDipole]], TotalCycles → 4, PointsPerCycle → 150]
```



You can specify a Target chemical species using the option

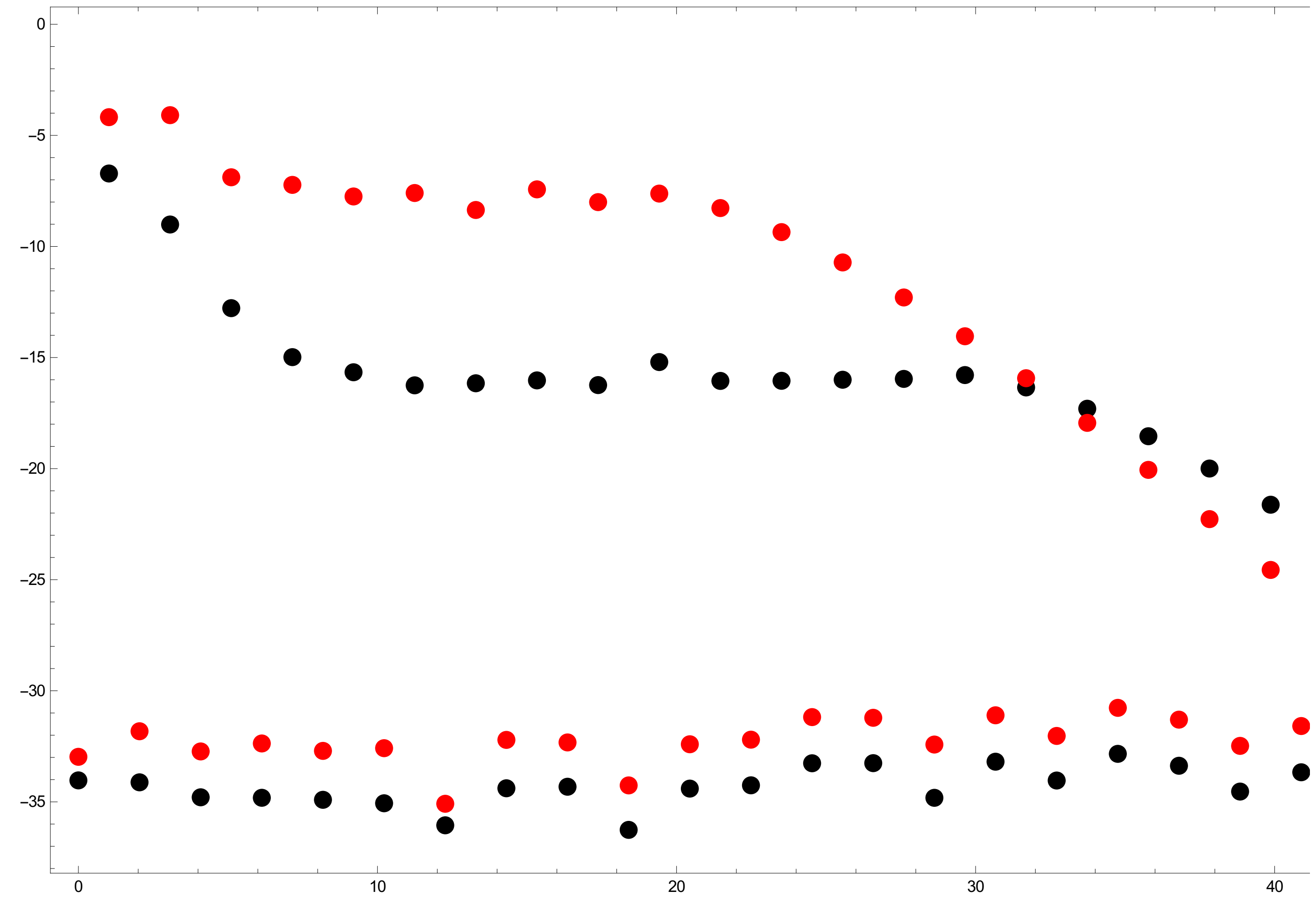
```
? Target
```

Target is an option for makeDipoleList which specifies chemical species producing the
HHG emission, pulling the ionization potential from the Wolfram ElementData curated data set.

i.e. using the syntax

```
AbsoluteTiming[
 heliumDipole = makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
   FieldParameters → {F → 0.05, ω → 0.057}, Target → "Helium"];
 xenonDipole = makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
   FieldParameters → {F → 0.05, ω → 0.057}, Target → "Xenon"];
]
{8.04619, Null}
```

```
Show[{
  spectrumPlotter[getSpectrum[Most[heliumDipole]], Joined → False, PlotStyle → Black],
  spectrumPlotter[getSpectrum[Most[xenonDipole]], Joined → False, PlotStyle → Red]
 }]
```



For convenience, the function getIonizationPotential gives a public-facing access to this functionality, via

```
? getIonizationPotential
```

getIonizationPotential[Target] returns the ionization potential of an atomic target, e.g. "Hydrogen", in atomic units.

getIonizationPotential[Target,q] returns the ionization potential of the q–th ion of the specified Target, in atomic units.

so that e.g.

```
{"H", #, UnitConvert[Quantity[#, "Hartrees"], "Electronvolts"]} &[
 getIonizationPotential["Hydrogen"]]
{"He+", #, UnitConvert[Quantity[#, "Hartrees"], "Electronvolts"]} &[
 getIonizationPotential["Helium", 1]]
```

```
{H, 0.49971, 13.598 eV}

{He+, 1.9998, 54.418 eV}
```

An ionization potential can also be specified directly:

```
? IonizationPotential
```

IonizationPotential is an option for makeDipoleList which specifies the ionization potential I_p of the target.

To see the available options for this function (and others), use

```
Options[makeDipoleList]
```

```
{PointsPerCycle → 90, TotalCycles → 1, CarrierFrequency → 0.057,
 VectorPotential → Automatic, FieldParameters → {}, VectorPotentialGradient → None,
 Preintegrals → Analytic, ReportingFunction → Identity, Gate → SineSquaredGate[1/2],
 nGate → 3/2, ϵCorrection → 0.1, IonizationPotential → 0.5,
 Target → Automatic, DipoleTransitionMatrixElement → hydrogenicDTME,
 PointNumberCorrection → 0, Verbose → 0, IntegrationPointsPerCycle → Automatic}
```

All options have suitable information messages.

```
? VectorPotential
```

VectorPotential is an option for makeDipole list which specifies the field's vector potential. Usage should be VectorPotential→A, where A[t]//.pars must yield a list of numbers for numeric t and parameters indicated by FieldParameters→pars.

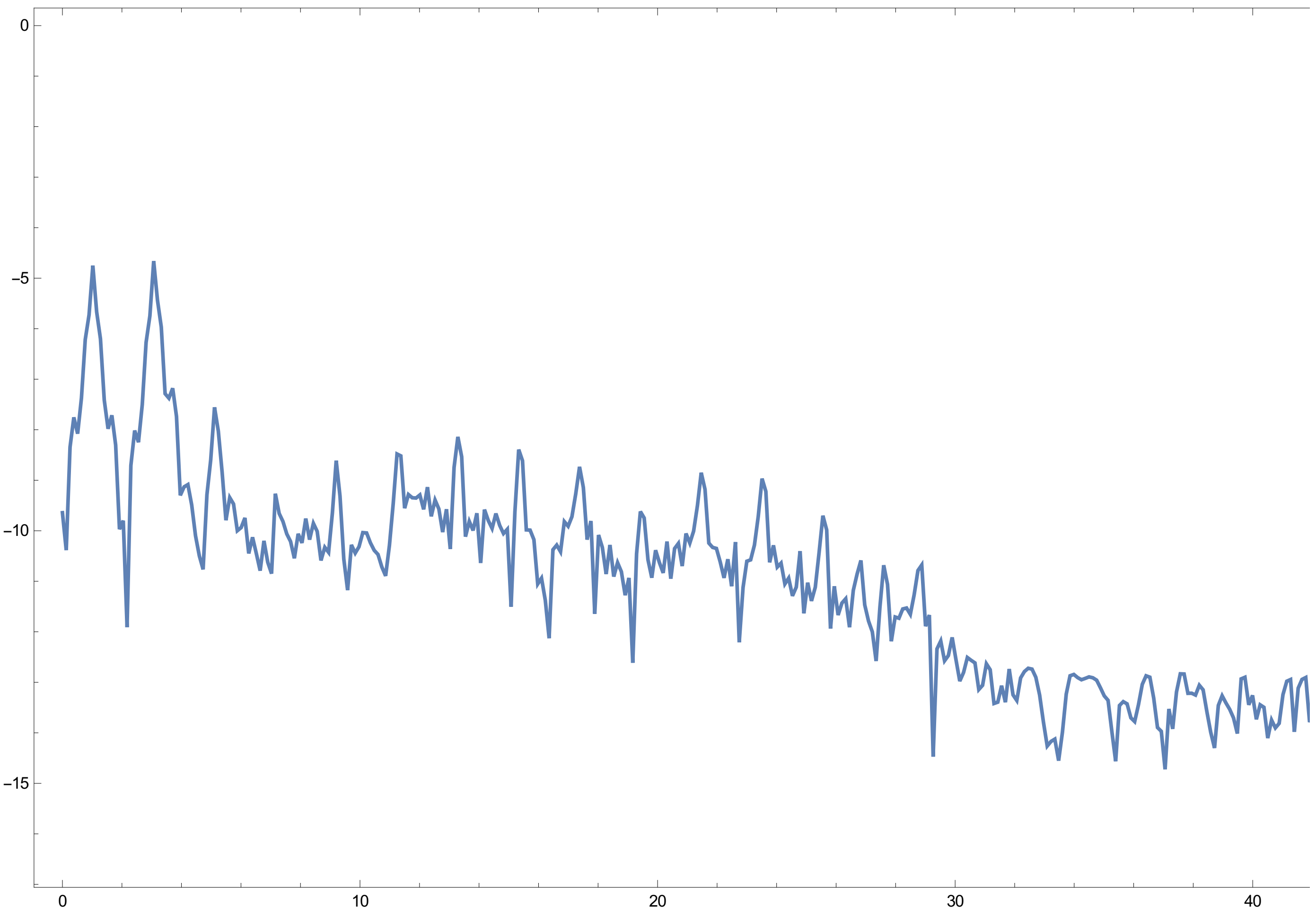
Using numerical integration for the preintegrals

Dipole case

To simulate a pulse with an envelope, it can be convenient to perform the preintegrals numerically, using the option Preintegrals→”Numeric”. These cases are generally slower but mainly because they require many more periods of integration.

```
AbsoluteTiming[
 numericallyIntegratedDipole =
   makeDipoleList[VectorPotential → Function[t, {F/ω envelope[t] Sin[ω t], 0, 0}]
     , FieldParameters → {ω → 0.057, F → 0.055, envelope → cosPowerFlatTop[0.057, 8, 16]}
     , TotalCycles → 8
     , Preintegrals → "Numeric"
    ];
]
{20.4918, Null}
```

```
spectrumPlotter[getSpectrum[numericallyIntegratedDipole]]
```



When using flat top pulses, and other waveforms that depend on `Piecewise` functions, it is possible that the function will return errors caused by an Indeterminate derivative being evaluated at the corners of the envelope.

```
AbsoluteTiming[
 flatTopPulseDipole =
   makeDipoleList[VectorPotential → Function[t, {F/ω envelope[t] Sin[ω t], 0, 0}],
    FieldParameters → {ω → 0.057, F → 0.055, envelope → flatTopEnvelope[0.057, 8, 2]},
    TotalCycles → 8, Preintegrals → "Numeric"];
]
```

```
{21.6107, Null}
```

In these cases, use a numeric test to diagnose what’s happened

```
Tally[flatTopPulseDipole /. _?NumberQ → ✓]
```

```
{{{✓, ✓, ✓}, 721}}
```

and if the function is returning non-numeric values, it can help to fiddle with the PointNumberCorrection option.

```
? PointNumberCorrection
```

PointNumberCorrection is an option for makeDipoleList and timeAxis
which specifies an extra number of points to be integrated over, which is useful to prevent
Indeterminate errors when a Piecewise envelope is being differentiated at the boundaries.

Nondipole case

The numerical Preintegrals can be used in the nondipole case but they’re obviously much slower. The number of preintegrals to find numerically increases from two in the dipole case ($\int \boldsymbol{A}(\tau)\,d\tau$ and $\int \boldsymbol{A}(\tau)^2\,d\tau$) to eight with the nondipole contributions, three of them parametrized by *t*'. The main load, however, is not in numerically calculating these integrals via NDSolve constructs, but rather in the added strain of accessing the preintegrals as Interpolating⋅. Function objects once they’ve been calculated, from the main integration loop.

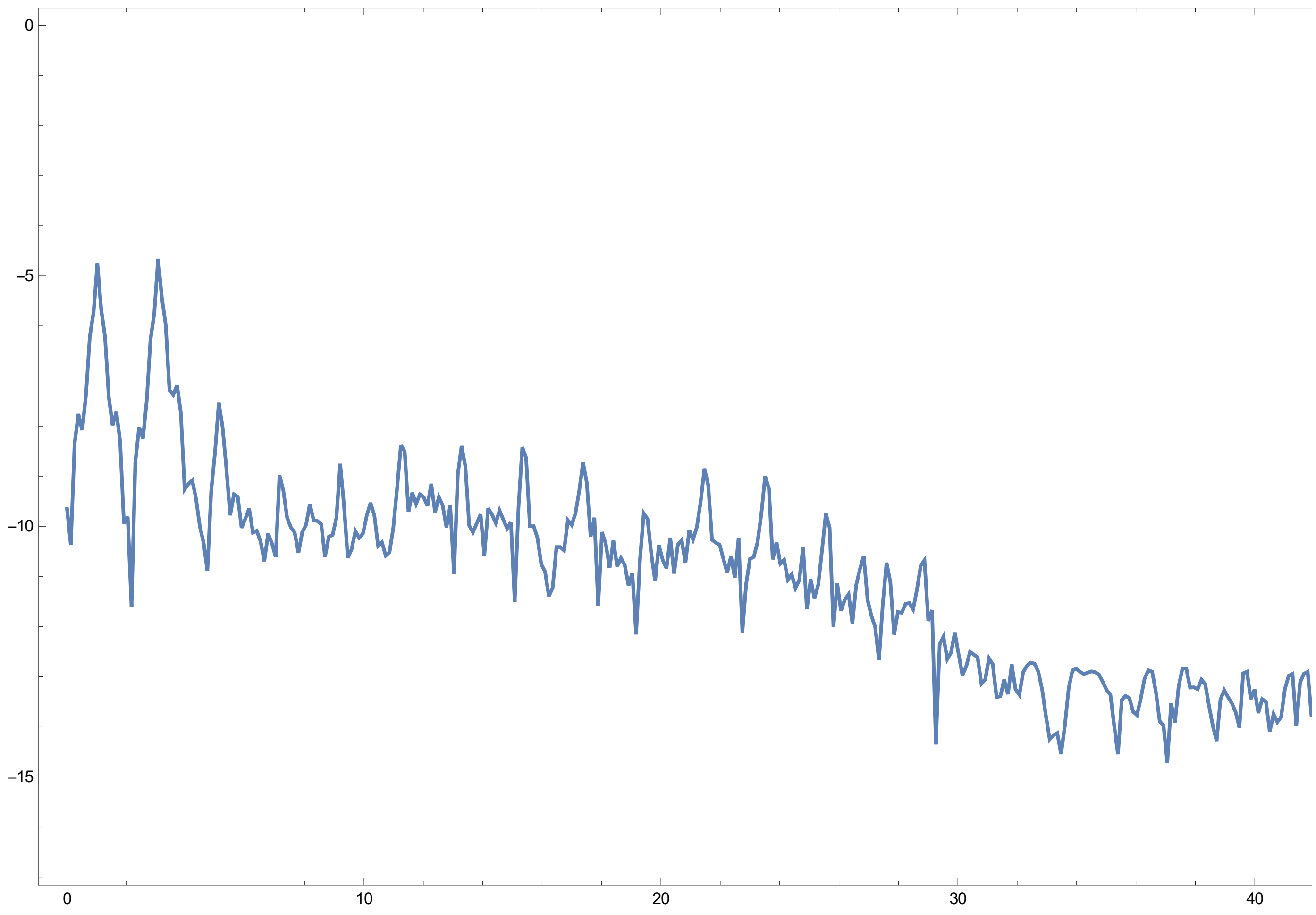
```
DateString[]
AbsoluteTiming[
 numericallyIntegratedNonDipoleCaseDipole = makeDipoleList[
    VectorPotential → Function[t, {F/ω envelope[t] Sin[ω t], 0, 0}],
    VectorPotentialGradient →
     Function[t, {{0, 0, 0}, {0, 0, 0}, {-k F/ω envelope[t] Sin[ω t], 0, 0}}],
    FieldParameters → {ω → 0.057, F → 0.055, envelope → cosPowerFlatTop[0.057, 8, 16],
      k → ω ω, α → 1 / 20}
    , TotalCycles → 8, Preintegrals → "Numeric"
   ];
]
DateString[]
Beep[]
```

```
Wed 17 Feb 2016 12:38:15
```

```
{73.411, Null}
```

```
Wed 17 Feb 2016 12:39:29
```

```
spectrumPlotter[getSpectrum[numericallyIntegratedNonDipoleCaseDipole]]
```



Parallelization

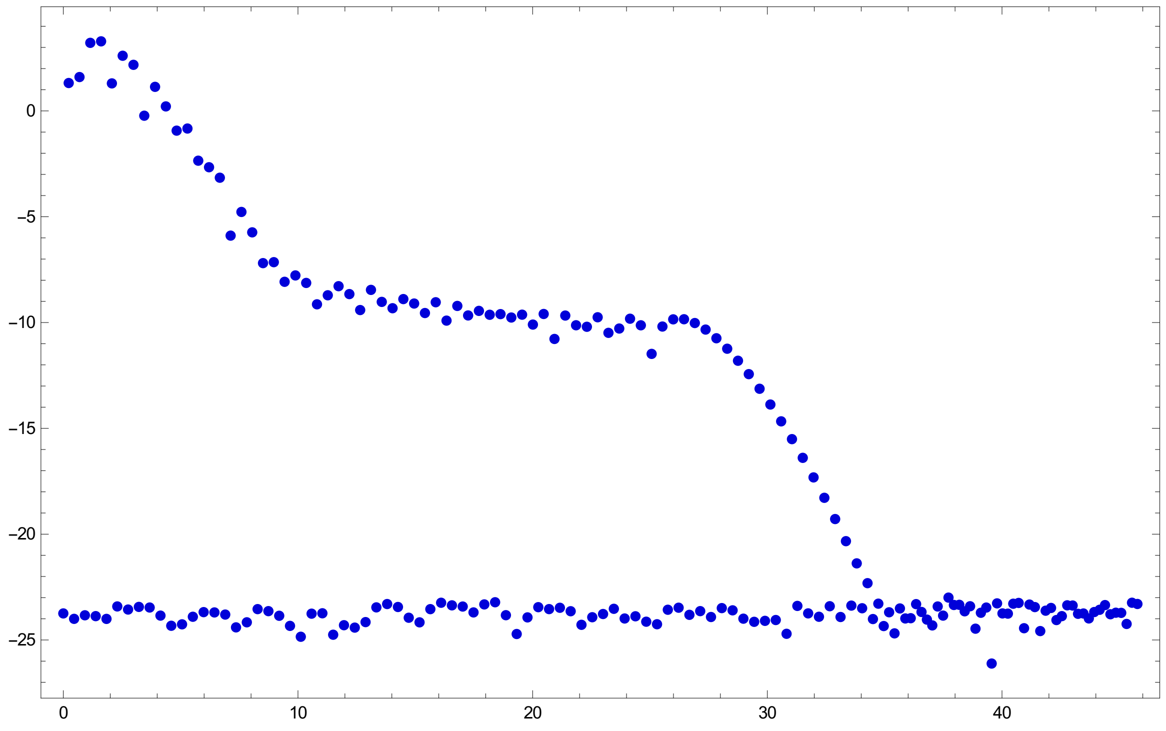
Parallelizing single instances

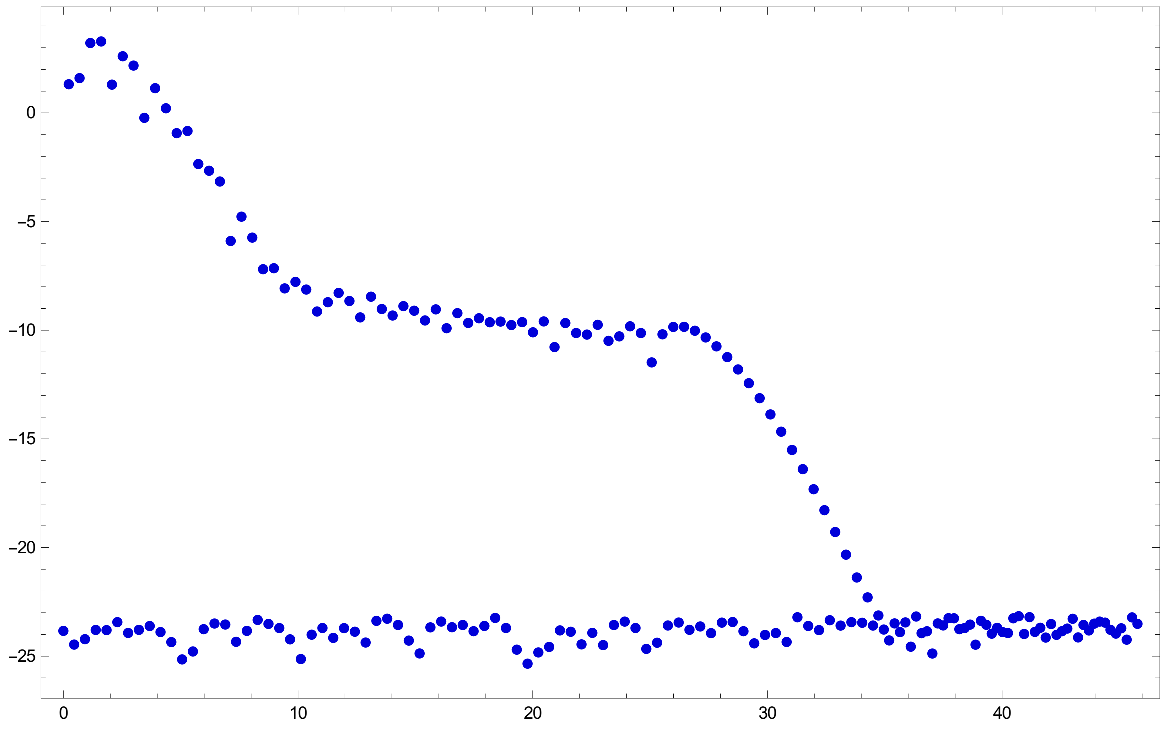
For faster evaluation of a single instance, it is possible to parallelize the evaluation, by adding the option RunInParallel → True.

```
AbsoluteTiming[directDipole = makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
    FieldParameters → {F → √10 0.05, ω → 0.057}, PointsPerCycle → 400, RunInParallel → False];]
AbsoluteTiming[parallelizedDipole = makeDipoleList[
    VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
    FieldParameters → {F → √10 0.05, ω → 0.057}, PointsPerCycle → 400, RunInParallel → True];]
```

```
{49.3113, Null}

{16.3488, Null}
```

```
Row[{
  spectrumPlotter[getSpectrum[directDipole⟦1 ;; -2⟧],
   Joined → False, PlotStyle → Darker[Blue, 0.15], ImageSize → 600],
  spectrumPlotter[getSpectrum[parallelizedDipole⟦1 ;; -2⟧],
   Joined → False, PlotStyle → Darker[Blue, 0.15], ImageSize → 600]
 }]
```





Unfortunately, the in-package single-instance parallelization can be unstable on occasion; this is probably due to a bug in ParallelTable (which can, under enough load, return different results to Table, in which case results typically differ run-to-run) that has proven so far very difficult to diagnose.

In such cases, the RunInParallel option takes a third possibility - an explicit set of commands, {TableCommand, SumCommand}, to use in the iteration.

```
? RunInParallel
```

RunInParallel is an option for makeDipoleList which controls whether each RB–SFA instance is parallelized. It accepts False as the (Automatic) option, True, to parallelize each instance, or a pair of functions {TableCommand, SumCommand} to use for the iteration and summing, which could be e.g. {Inactive[ParallelTable], Inactive[Sum]}.

This is meant to be used by changing those commands to dud versions which can be sprung up later. The ideal use case (in v10 and up) is via Inactive commands, which return as

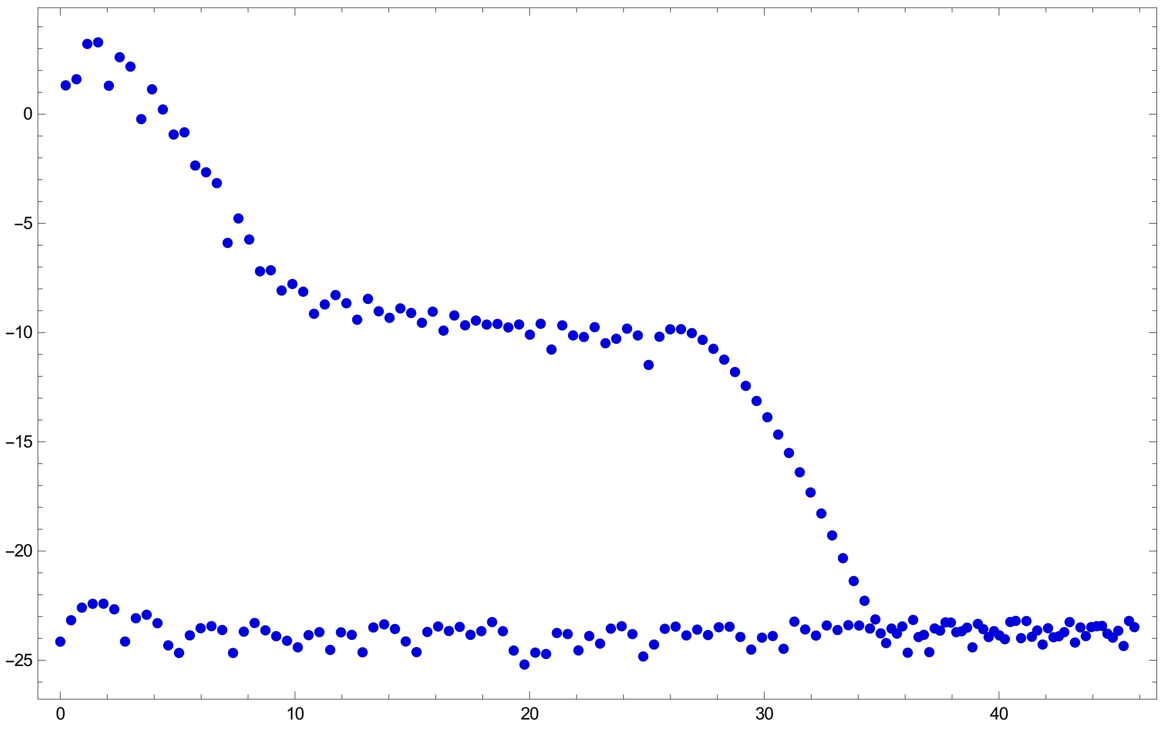
```
makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
  FieldParameters → {F → √10 0.05, ω → 0.057}, Gate → (1 &), PointsPerCycle → 400
  , RunInParallel → {Inactive[ParallelTable], Inactive[Sum]}
 ] /. {RBSFA`Private`t → t, RBSFA`Private`τ → τ}
```

```
ParallelTable[0.275578 Sum[{((56.7185 + 0. ⅈ)
    ⅇ^(-ⅈ (-(-48.6654 Cos[0.057 t]+48.6654 Cos[0.057 (t-τ)])^2/((0.-0.1 ⅈ)+τ) + 1/2 τ (1.+(-48.6654 Cos[0.057 t]+48.6654 Cos[0.057 (t-τ)])^2/((0.-0.1 ⅈ)+τ)^2) + 1/2 (7.69468 (0.5 t-4.38596 Sin[0.114 
     (1/(0.1 + ⅈ τ))^(3/2) (-((-48.6654 Cos[0.057 t] + 48.6654 Cos[0.057 (t - τ)]) /
          ((0. - 0.1 ⅈ) + τ)) + 2.77393 Sin[0.057 t])
     (0. - ((0. + 0.56941 ⅈ) Cos[0.057 (t - τ)] (-((-48.6654 Cos[0.057 t] + 48.6654
                 Cos[0.057 (t - τ)]) / ((0. - 0.1 ⅈ) + τ)) + 2.77393 Sin[0.057 (t - τ)])) /
         (1. + (-((-48.6654 Cos[0.057 t] + 48.6654 Cos[0.057 (t - τ)]) / ((0. - 0.1 ⅈ) + τ)) +
               2.77393 Sin[0.057 (t - τ)])^2)^3)) /
   (1. + (-((-48.6654 Cos[0.057 t] + 48.6654 Cos[0.057 (t - τ)]) / ((0. - 0.1 ⅈ) + τ)) +
          2.77393 Sin[0.057 t])^2)^3,
  (0. + 0. ⅈ) (1/(0.1 + ⅈ τ))^(3/2) (0. - ((0. + 0.56941 ⅈ) Cos[0.057 (t - τ)]
        (-((-48.6654 Cos[0.057 t] + 48.6654 Cos[0.057 (t - τ)]) / ((0. - 0.1 ⅈ) + τ)) +
          2.77393 Sin[0.057 (t - τ)])) /
       (1. + (-((-48.6654 Cos[0.057 t] + 48.6654 Cos[0.057 (t - τ)]) / ((0. - 0.1 ⅈ) + τ)) +
             2.77393 Sin[0.057 (t - τ)])^2)^3),
  (0. + 0. ⅈ) (1/(0.1 + ⅈ τ))^(3/2) (0. - ((0. + 0.56941 ⅈ) Cos[0.057 (t - τ)]
        (-((-48.6654 Cos[0.057 t] + 48.6654 Cos[0.057 (t - τ)]) / ((0. - 0.1 ⅈ) + τ)) +
          2.77393 Sin[0.057 (t - τ)])) /
       (1. + (-((-48.6654 Cos[0.057 t] + 48.6654 Cos[0.057 (t - τ)]) / ((0. - 0.1 ⅈ) + τ)) +
             2.77393 Sin[0.057 (t - τ)])^2)^3)},
 {τ, 0, 165.347, 0.275578}], {t, 0, 110.231, 0.275578}]
```

and which can then be sprung into action using Activate:

```
AbsoluteTiming[postActivatedDipole = Activate[
    makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
     FieldParameters → {F → √10 0.05, ω → 0.057}, PointsPerCycle → 400,
     RunInParallel → {Inactive[ParallelTable], Inactive[Sum]}]
   ];]
spectrumPlotter[getSpectrum[postActivatedDipole[[1 ;; -2]]],
 Joined → False, PlotStyle → Darker[Blue, 0.15], ImageSize → 600]
```

```
{5.95945, Null}
```



This looks like it shouldn’t change anything, but it can help fix a noisy ParallelTable output. It can also, inexplicably, be rather faster than the in-package parallelization. (For a cleaner example of the latter, see this mathematica.stackexchange question.)

Multiple instances in parallel

Alternatively, one can also parallelize over each run by using ParallelTable and similar commands. In general, this requires some careful handling of contexts; in this package this has been resolved by including the package context into the $DistributedContexts variable via the assignment

```
$DistributedContexts := {$Context, "RBSFA`"}
```

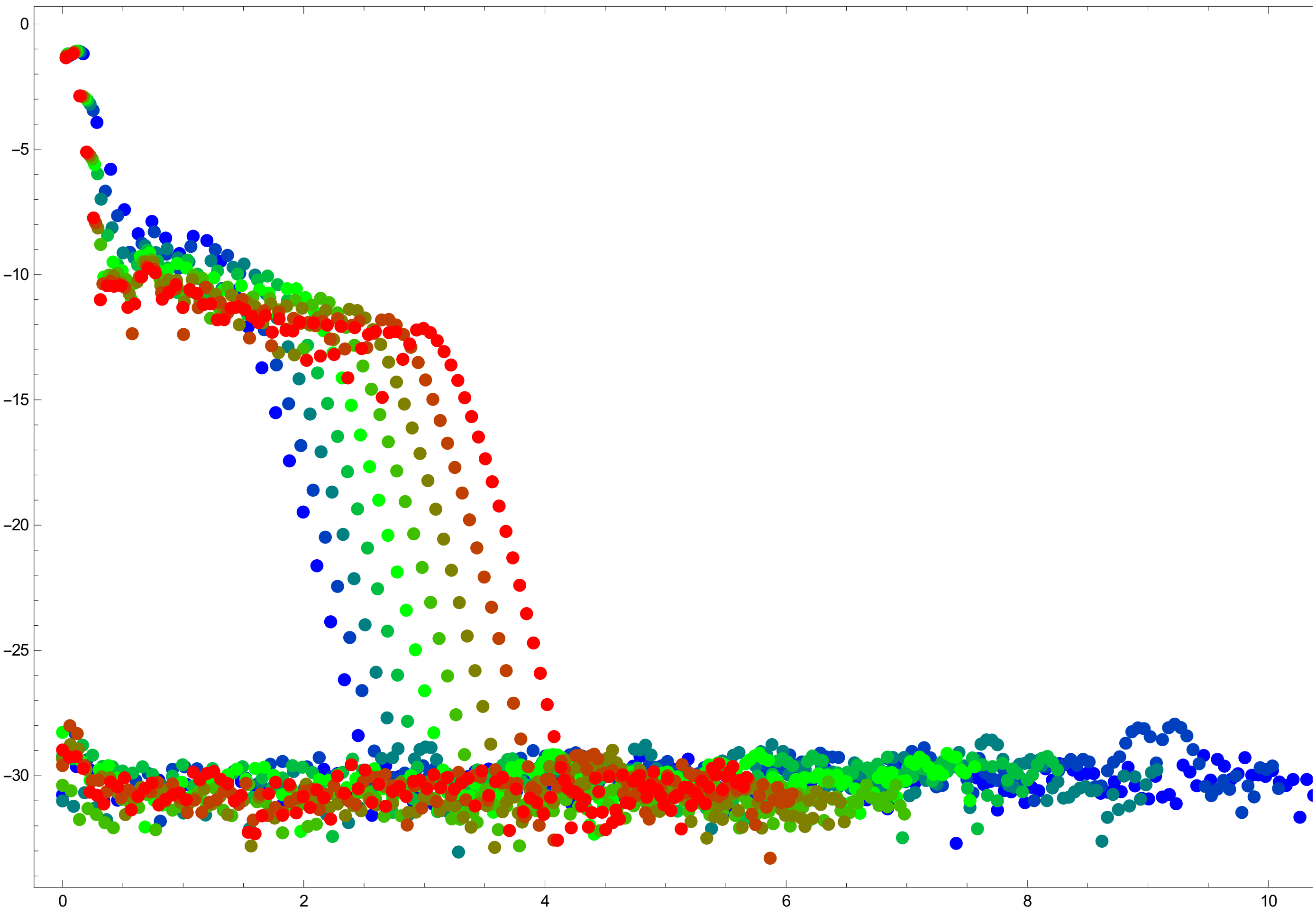
as opposed to the standard $DistributedContexts:=$Context setting. (This means, though, that loading this package can break other stuff if you’ve set $DistributedContexts to something other than the default; if this is the case you will get a warning on $DistributedContexts::overwrite when loading the package.)

In addition to this, if a variable or function is used to store the results, this must be synchronized using SetShared Function or SetSharedVariable, as usual.

```
SetSharedFunction[wavelengthScanDipole];

ParallelTable[
 Print[AbsoluteTiming[
   wavelengthScanDipole[λ] =
     makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}], FieldParameters →
       {F → 0.05, ω → 45.6 / λ}, CarrierFrequency → 45.6 / λ, PointsPerCycle → 400];
   ]]
 , {λ, 800, 1600, 100}]
```

```
{55.5889, Null}

{55.6341, Null}

{55.7321, Null}

{55.7603, Null}

{54.8449, Null}

{56.393, Null}

{57.5908, Null}

{57.4291, Null}

{51.2366, Null}

{Null, Null, Null, Null, Null, Null, Null, Null, Null}
```

```
Show[Table[
  spectrumPlotter[getSpectrum[Most[wavelengthScanDipole[λ]]],
   PlotStyle → Blend[{Blue, Green, Red}, λ / 800 - 1], CarrierFrequency → 45.6 / λ,
   Joined → False, FrequencyAxis → "Frequency", PointsPerCycle → 400]
  , {λ, 800, 1600, 100}]]
```



Writing output to file

For very large calculations (many integration points per cycle, in particular), the limiting factor is available memory. In these situations, it can help to write the data directly to a file on disk. This is slower (by a factor of about 2) but it has a roughly constant RAM footprint, so it enables calculations of a bigger size than would be possible otherwise. (Of course, this can also be done from non-parallelized calls!) This is done via the ReportingFunction option:

```
? ReportingFunction
```

ReportingFunction is an option for makeDipole list which specifies a function
 used to report the results, either internally (by the default, Identity) or to an external file.

In essence, the integration loop consists of a Table construct, which goes over the time *t* at which the integral is performed, and an inner integration construct. Setting an option ReportingFunction→f interposes the function f between these two steps, as

```
Table[ f[ integrator[t] ] , {t, tInitial, tFinal}]
```

The default is f=Identity, which returns its input untouched, but it can also be replaced by a Write construct that can shunt its input to the hard disk without telling the kernel what it is, so it is not kept in memory.

```
Quit
```

```
DistributeDefinitions["RBSFA`"];
directory = NotebookDirectory[];
filename[F_] := FileNameJoin[{directory, "Field scan data at F=" <> ToString[F] <> ".txt"}];
```

```
ParallelTable[
 Print[AbsoluteTiming[
   makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
     FieldParameters → {ω → 0.057}, CarrierFrequency → 0.057, PointsPerCycle → 400,
     ReportingFunction → Function[Write[filename[F], #]]
    ];
  ]]
 , {F, 0.05, 0.2, 0.025}]
```

```
{66.2765, Null}

{68.2327, Null}

{68.4573, Null}

{69.0505, Null}

{69.6457, Null}

{69.8211, Null}

{70.4778, Null}

{Null, Null, Null, Null, Null, Null, Null}
```

The data in the files can then be pulled in quite simply using e.g.

```
Do[ intensityScanDipole[F] = ReadList[filename[F]], {F, 0.05, 0.2, 0.025}]
```

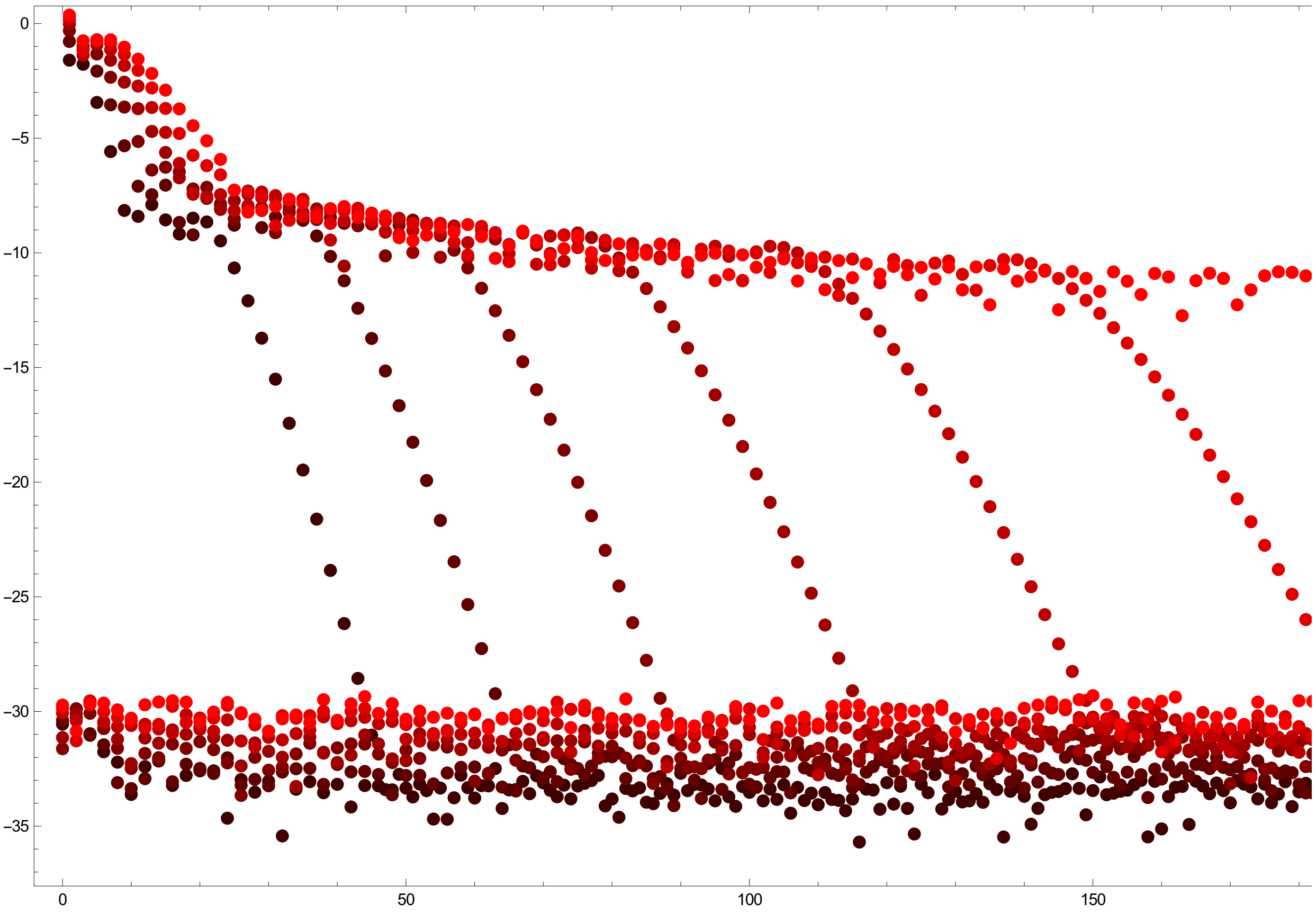
This tends to litter the directories by creating lots of files for different parameters, so it is usually cleaner to Save them into a single file, e.g. using

```
Save[FileNameJoin[{NotebookDirectory[], "Field scan collected data.txt"}],
 intensityScanDipole]
```

which in turn can then be pulled in using

```
<< (FileNameJoin[{NotebookDirectory[], "Field scan collected data.txt"}]);
```

```
Show[Table[
  spectrumPlotter[getSpectrum[Most[intensityScanDipole[F]], CarrierFrequency → 0.057,
   Joined → False, PointsPerCycle → 400, PlotStyle → Blend[{Black, Red}, F / 0.2]]
  , {F, 0.05, 0.2, 0.025}]]
```



As written, though, this has the disadvantage that each subkernel must access the hard drive for *every* timestep of the computation, which obviously responsible for (at least most of) the slowdown. A middle ground is also possible by choosing an appropriate ReportingFunction: a function which will cache a specific number *k* of results on RAM, and then write them to file all in one go. This is on the development to do (wish) list, and will hopefully be implemented soon - if time allows.

Time and memory use

Benchmark evaluation

The benchmarks below were taken on a desktop machine with 8-thread, 4-core Intel i7-3770 CPU at 3.40GHz, 16GB RAM, running *Mathematica* 10.0.1 over Ubuntu 14.04. The time taken per computation depends most strongly on the PointsPerCycle used to sample and integrate, and the dependence is therefore quadratic.

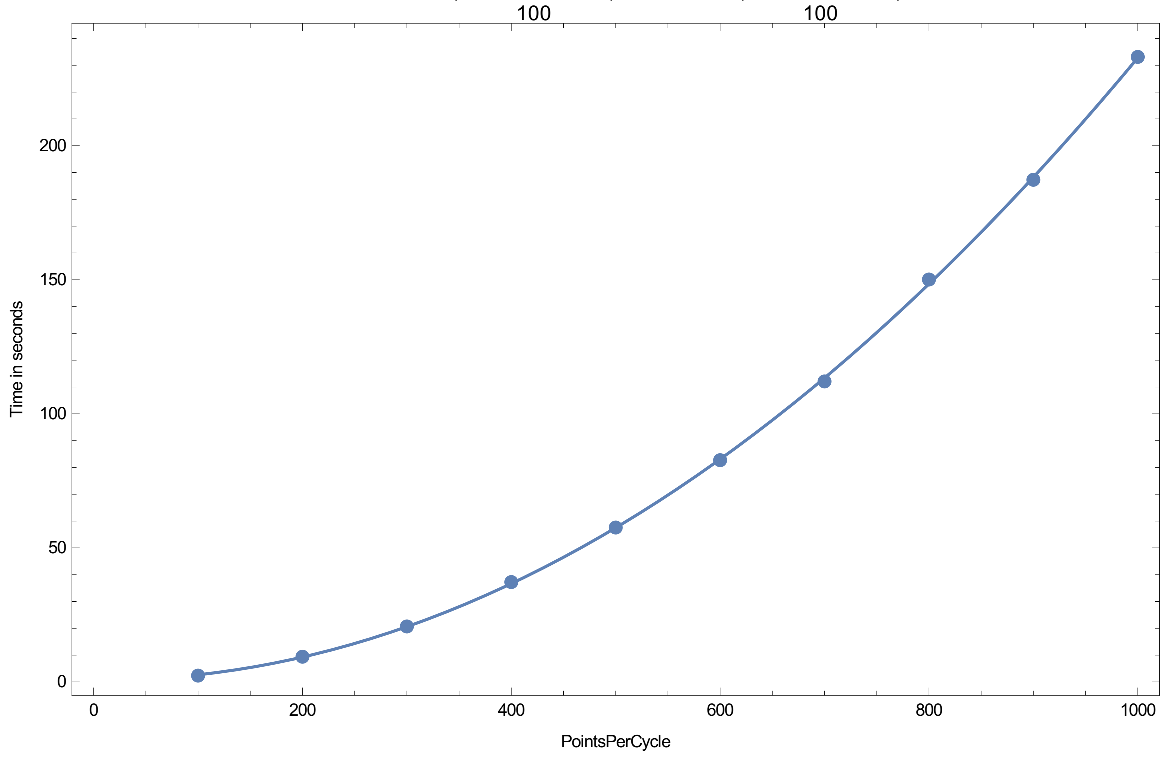
```
timingsList = Table[
  {n, AbsoluteTiming[
    MaxMemoryUsed[makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
      FieldParameters → {F → √(n/100) 0.053, ω → 0.057}, PointsPerCycle → n]]]}
  ,
  {n,
   100,
   1000,
   100}]
```

```
{{100, {2.36167, 4 905 296}}, {200, {9.40602, 19 018 888}},
 {300, {20.6878, 43 497 000}}, {400, {37.2461, 78 976 528}}, {500, {57.5847, 118 770 848}},
 {600, {82.7101, 173 233 496}}, {700, {112.082, 232 621 352}}, {800, {150.096, 301 125 912}},
 {900, {187.298, 387 140 808}}, {1000, {233.122, 473 885 368}}}
```

Timings

```
timingsModel = LinearModelFit[(Flatten /@ timingsList)⟦All, {1, 2}⟧, {1, n, n^2}, {n}];
Show[{
  ListPlot[
   (Flatten /@ timingsList)⟦All, {1, 2}⟧
  ],
  Plot[timingsModel[n], {n, timingsList⟦1, 1⟧, timingsList⟦-1, 1⟧}]
 }
 , Frame → True, PlotLabel → Row[{"time in seconds=", timingsModel[100 "(PointsPerCycle/100)"]}],
 FrameLabel → {"PointsPerCycle", "Time in seconds"}, ImageSize → 600
]
```

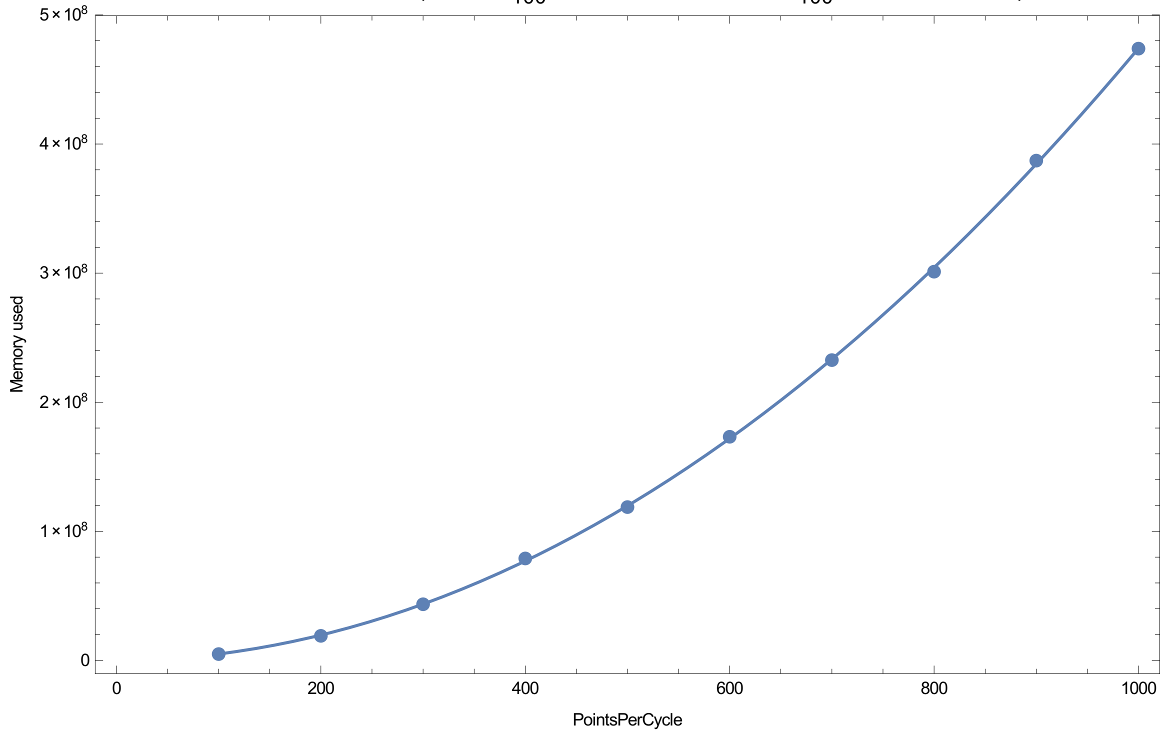
time in seconds=2.3726 $\left(\frac{\text{PointsPerCycle}}{100}\right)^2 - 0.530343 \left(\frac{\text{PointsPerCycle}}{100}\right) + 0.831196$



Maximum memory used

```
memoryModel = LinearModelFit[(Flatten /@ timingsList)[[All, {1, 3}]], {1, n, n^2}, {n}];
Show[{
  ListPlot[
   (Flatten /@ timingsList)[[All, {1, 3}]]
  ],
  Plot[memoryModel[n], {n, timingsList[[1, 1]], timingsList[[-1, 1]]}]
 }
 , Frame → True,
 PlotLabel → Row[{"Memory used=", Simplify[memoryModel[100. "(PointsPerCycle/100)"] 10^-6 "MB"]}],
 FrameLabel → {"PointsPerCycle", "Memory used"}, ImageSize → 600
]
```

Memory used=4.68799 (1. (PointsPerCycle/100)^2 + 0.119601 (PointsPerCycle/100) − 0.0541501) MB



In parallel

Inside parallel environments the timings are somewhat slower, by a factor of about 1.8. The timings below were taken with 7 *Mathematica* kernels running in parallel.

```
DistributeDefinitions["RBSFA`"];
```

```
parallelTimingsList = ParallelTable[
  {n, AbsoluteTiming[MaxMemoryUsed[
     makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}], FieldParameters →
        {F → √(n/100) 0.053, ω → 45.6 / λ}, CarrierFrequency → 45.6 / λ, PointsPerCycle → n]]]}
  , {λ, 770, 830, 10}, {n, 100, 1000, 100}]
```

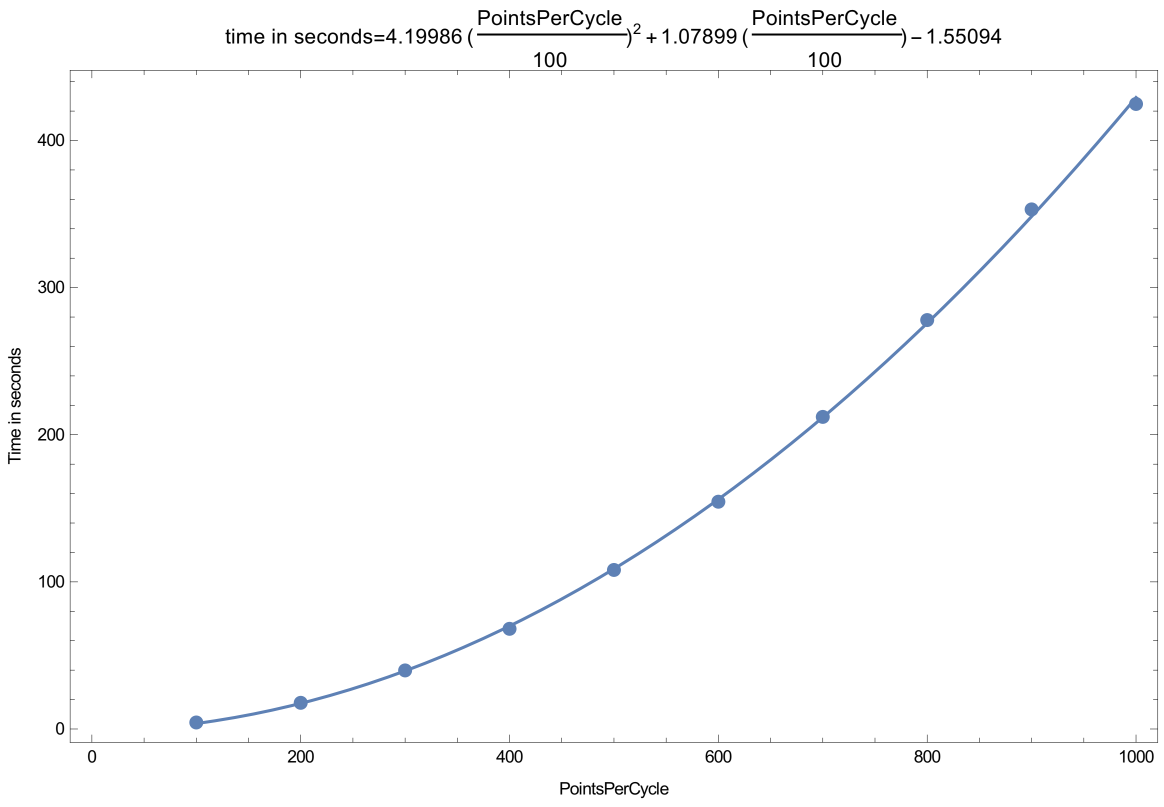
```
{{{100, {4.71833, 7 949 312}}, {200, {18.2567, 19 028 920}},
  {300, {39.6818, 43 507 248}}, {400, {68.0825, 75 529 984}}, {500, {108.27, 118 772 272}},
  {600, {154.871, 173 234 856}}, {700, {211.182, 232 622 488}}, {800, {275.072, 301 126 208}},
  {900, {350.061, 387 140 984}}, {1000, {424.31, 473 885 664}}},
 {{100, {3.7169, 7 949 264}}, {200, {17.4824, 19 028 912}}, {300, {40.1556, 43 507 248}},
  {400, {68.1348, 75 529 984}}, {500, {109.399, 118 772 272}},
  {600, {153.631, 173 234 976}}, {700, {211.956, 232 622 248}},
  {800, {279.29, 301 126 208}}, {900, {353.1, 387 141 104}}, {1000, {429.039, 473 885 664}}},
 {{100, {4.70831, 7 949 264}}, {200, {17.3451, 19 028 912}}, {300, {40.1826, 43 507 248}},
  {400, {68.9951, 75 529 984}}, {500, {109.87, 118 772 272}}, {600, {156.143, 173 234 976}},
  {700, {209.361, 232 622 488}}, {800, {279.29, 301 126 208}},
  {900, {350.784, 387 141 104}}, {1000, {418.914, 473 885 664}}},
 {{100, {4.27991, 7 949 264}}, {200, {18.6273, 19 028 912}}, {300, {40.4553, 43 507 248}},
  {400, {67.5833, 75 529 984}}, {500, {107.493, 118 772 272}}, {600, {151.738, 173 234 976}},
  {700, {211.565, 232 622 368}}, {800, {276.237, 301 126 208}},
  {900, {352.376, 387 141 104}}, {1000, {429.842, 473 885 664}}},
 {{100, {4.80072, 7 949 264}}, {200, {17.9797, 19 028 912}}, {300, {38.2944, 43 507 248}},
  {400, {68.7844, 75 529 984}}, {500, {105.406, 118 772 272}}, {600, {151.986, 173 234 976}},
  {700, {216.018, 232 622 488}}, {800, {279.116, 301 126 208}},
  {900, {361.033, 387 141 104}}, {1000, {424.589, 473 885 664}}},
 {{100, {4.24956, 7 949 264}}, {200, {17.1782, 19 028 912}}, {300, {40.1991, 43 507 248}},
  {400, {67.2693, 75 529 864}}, {500, {108.852, 118 772 032}},
  {600, {156.147, 173 234 736}}, {700, {210.96, 232 622 008}}, {800, {279.971, 301 126 208}},
  {900, {352.399, 387 141 104}}, {1000, {423.345, 473 885 664}}},
 {{100, {4.4809, 7 949 264}}, {200, {17.6272, 19 028 912}}, {300, {39.652, 43 507 248}},
  {400, {67.7538, 75 529 984}}, {500, {107.413, 118 772 272}}, {600, {156.833, 173 234 976}},
  {700, {213.902, 232 622 488}}, {800, {276.711, 301 126 208}},
  {900, {352.447, 387 140 984}}, {1000, {423.858, 473 885 544}}}}
```

```
parallelTimingsListAveraged =
 Table[{parallelTimingsListᵀ⟦k, 1, 1⟧, Mean[ parallelTimingsListᵀ⟦k, All, 2⟧]},
  {k, Length[parallelTimingsListᵀ]}]
```

```
{{100, {4.42209, 55 644 896/7}}, {200, {17.7852, 133 202 392/7}},
 {300, {39.803, 43 507 248}}, {400, {68.0862, 528 709 768/7}}, {500, {108.101, 831 405 664/7}},
 {600, {154.479, 1 212 644 472/7}}, {700, {212.135, 232 622 368}}, {800, {277.955, 301 126 208}},
 {900, {353.171, 2 709 987 488/7}}, {1000, {424.842, 3 317 199 528/7}}}
```

Timings

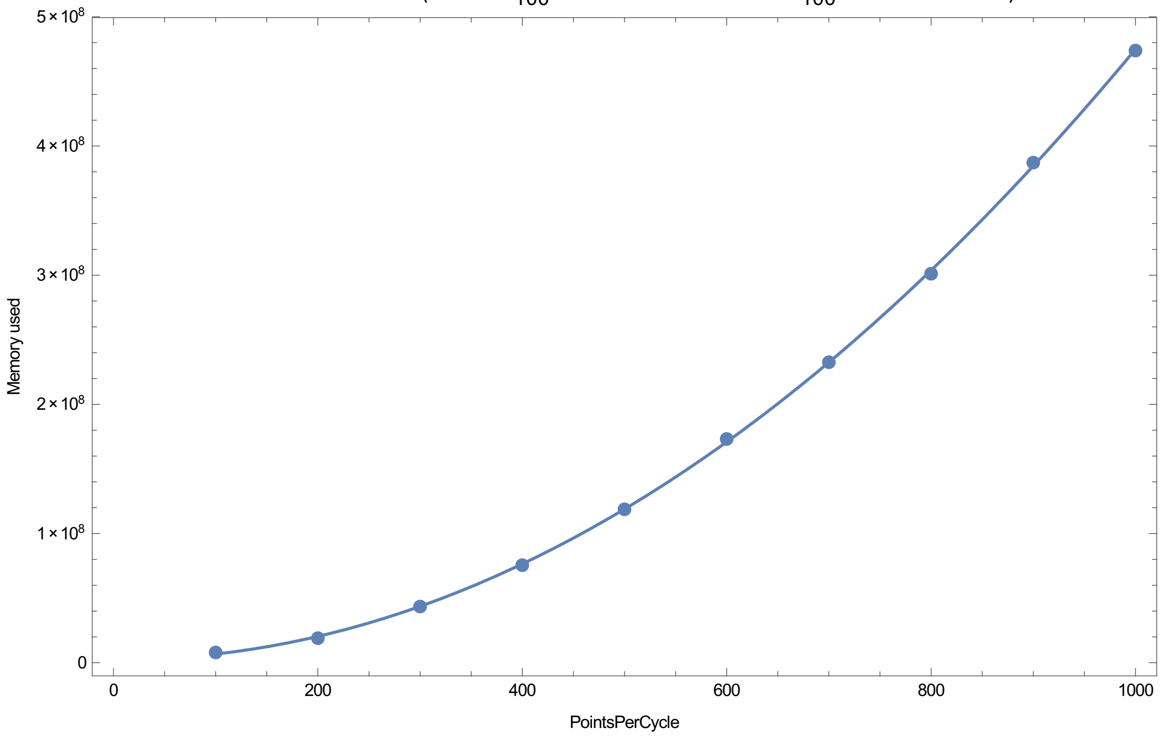
```
parallelTimingsModel =
  LinearModelFit[(Flatten /@ parallelTimingsListAveraged)[[All, {1, 2}]], {1, n, n^2}, {n}];
Show[{
  ListPlot[
   (Flatten /@ parallelTimingsListAveraged)[[All, {1, 2}]]
  ],
  Plot[parallelTimingsModel[n],
   {n, parallelTimingsListAveraged[[1, 1]], parallelTimingsListAveraged[[-1, 1]]}]
 }
 , Frame → True,
 PlotLabel → Row[{"time in seconds=", parallelTimingsModel[100 "(PointsPerCycle/100)"]}],
 FrameLabel → {"PointsPerCycle", "Time in seconds"}, ImageSize → 600
]
```



Memory

```
parallelMemoryModel =
  LinearModelFit[(Flatten /@parallelTimingsListAveraged)[[All, {1, 3}]], {1, n, n^2}, {n}];
Show[{
  ListPlot[
   (Flatten /@parallelTimingsListAveraged)[[All, {1, 3}]]
  ],
  Plot[parallelMemoryModel[n],
   {n, parallelTimingsListAveraged[[1, 1]], parallelTimingsListAveraged[[-1, 1]]}]
 }
 , Frame → True, PlotLabel →
  Row[{"Memory used=", Simplify[parallelMemoryModel[100. "(PointsPerCycle/100)"] 10^-6 "MB"]}],
 FrameLabel → {"PointsPerCycle", "Memory used"}, ImageSize → 600
]
```

Memory used=4.79633 $\left(1. \left(\frac{\text{PointsPerCycle}}{100}\right)^2 - 0.153257 \left(\frac{\text{PointsPerCycle}}{100}\right) + 0.555441\right)$ MB



Decoupling integration and sampling

If the quadratic scaling with respect to PointsPerCycle becomes too onerous, the sampling rate for the evaluation and for the numerical integration can be decoupled by providing an explicit option for IntegrationPointsPerCycle, which will set how many points are used per cycle in the numerical integration.

```
? IntegrationPointsPerCycle
```

IntegrationPointsPerCycle is an option for makeDipoleList which controls the number of points per cycle to use for the integration. Set to Automatic, to follow PointsPerCycle, or to an integer.

Obviously, if this option is taken, then the outcome should be checked for numerical convergence with respect to IntegrationPointsPerCycle.

```
DateString[]
decoupledTimingsList = Table[
  {nSampling, nIntegration, AbsoluteTiming[
    MaxMemoryUsed[makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}]
      , FieldParameters → {F → Sqrt[nSampling/100] 0.053, ω → 0.057}
      , PointsPerCycle → nSampling, IntegrationPointsPerCycle → nIntegration
     ]]]}
  , {nSampling, 100, 500, 100}, {nIntegration, 100, 500, 100}]
DateString[]
```

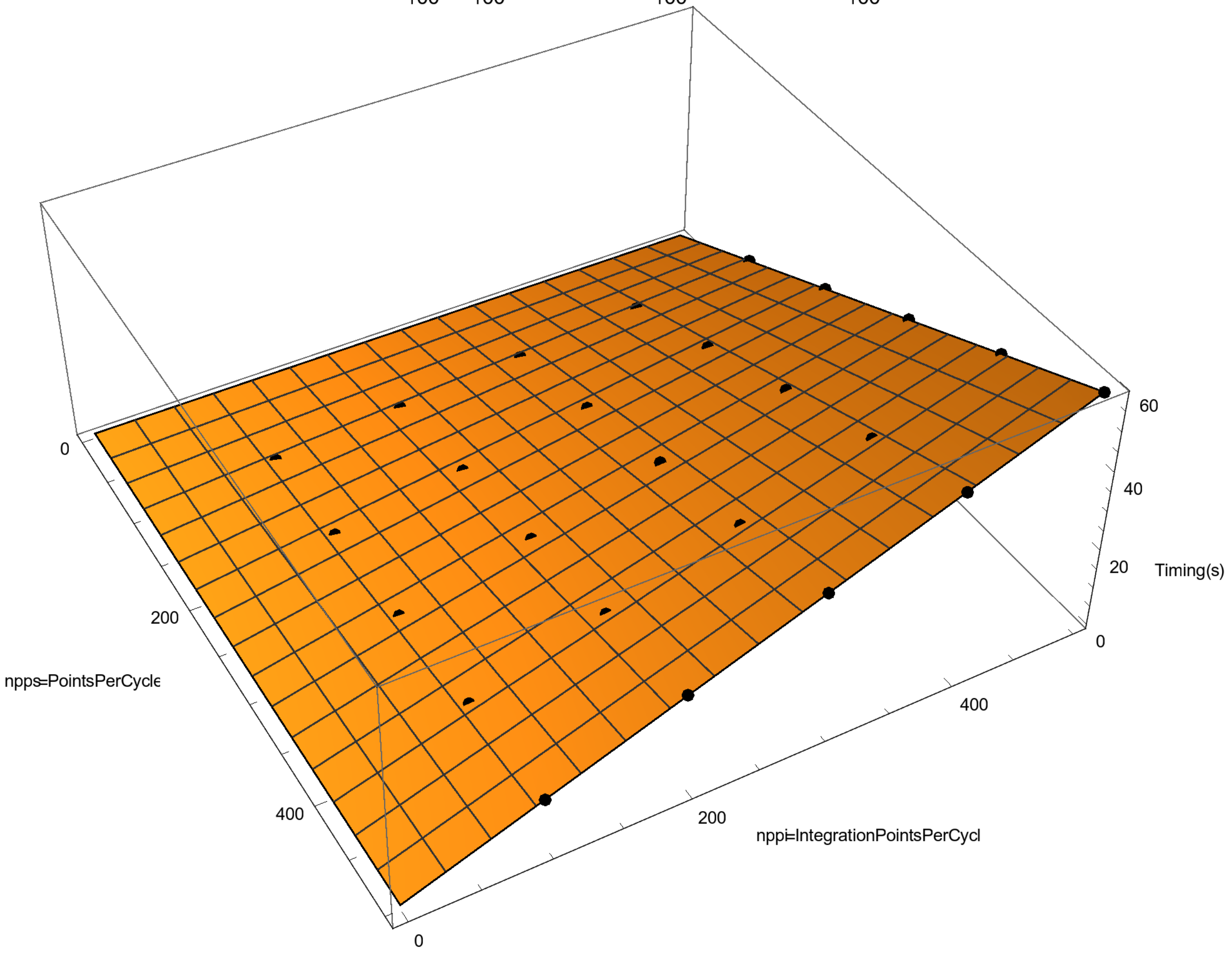
```
Fri 5 Feb 2016 17:08:36
```

```
{{{100, 100, {2.39139, 4 789 112}},
  {100, 200, {4.76862, 9 473 168}}, {100, 300, {7.07336, 14 241 584}},
  {100, 400, {9.39992, 19 138 656}}, {100, 500, {11.7964, 24 035 936}}},
 {{200, 100, {4.72937, 9 447 040}}, {200, 200, {9.3758, 19 017 072}},
  {200, 300, {14.1647, 28 478 336}}, {200, 400, {18.8825, 38 195 832}},
  {200, 500, {23.4484, 47 916 856}}}, {{300, 100, {7.02226, 14 177 120}},
  {300, 200, {14.067, 28 434 560}}, {300, 300, {21.5271, 43 233 320}},
  {300, 400, {28.4153, 56 993 480}}, {300, 500, {35.2399, 70 747 672}}},
 {{400, 100, {9.41508, 19 027 632}}, {400, 200, {18.6957, 38 108 104}},
  {400, 300, {28.159, 56 949 288}}, {400, 400, {37.5268, 75 259 432}},
  {400, 500, {46.83, 95 676 600}}}, {{500, 100, {11.6672, 23 882 240}},
  {500, 200, {23.5225, 47 786 456}}, {500, 300, {35.0966, 70 660 928}},
  {500, 400, {46.7308, 95 634 048}}, {500, 500, {58.5078, 118 501 720}}}}
```

```
Fri 5 Feb 2016 17:17:24
```

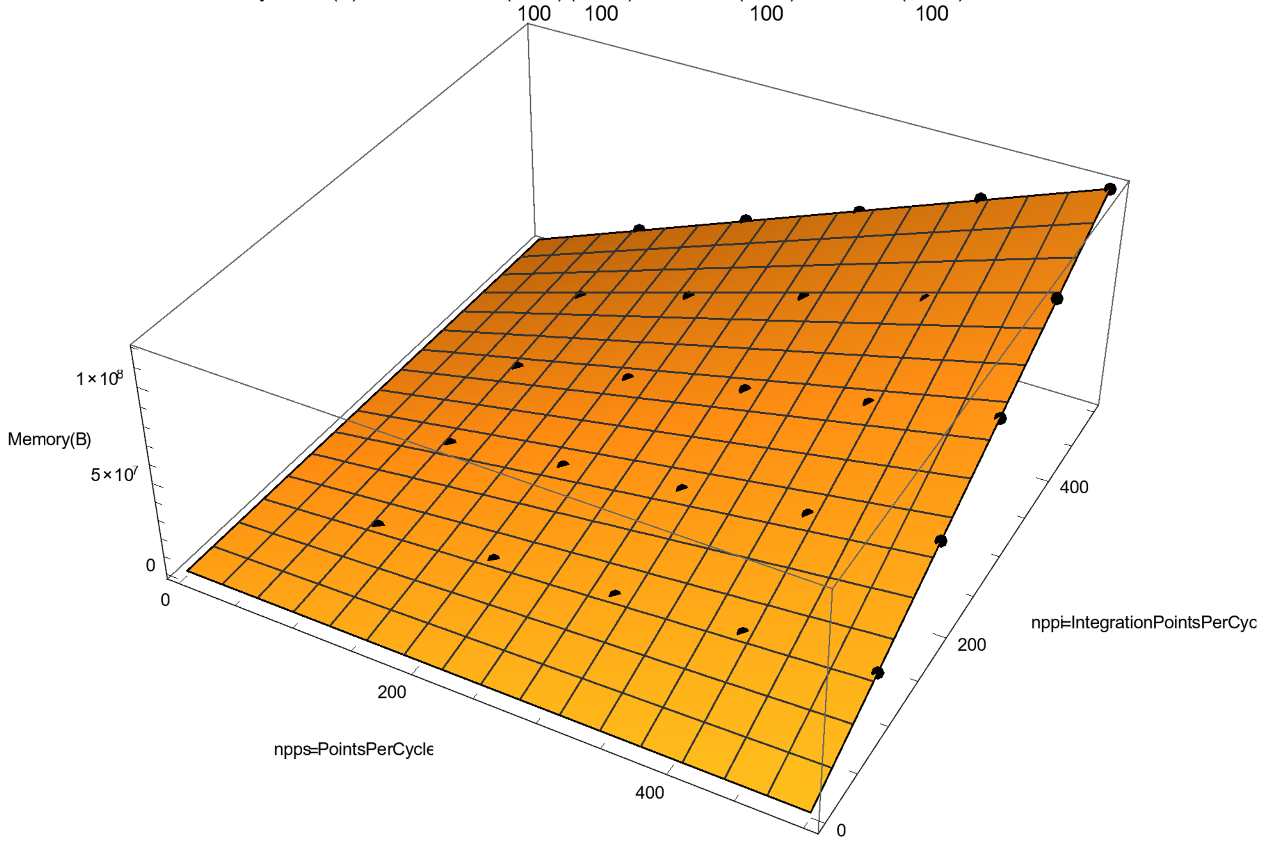
```
decoupledTimingsModel = LinearModelFit[
   (Flatten /@ Flatten[decoupledTimingsList, {{1, 2}}]) [[All, {1, 2, 3}]]
   , {1, nSampling, nIntegration, nSampling × nIntegration}, {nSampling, nIntegration}];
Show[{
  Plot3D[
   decoupledTimingsModel[nSampling, nIntegration]
   , {nSampling, 0, 500}, {nIntegration, 0, 500}
   , AxesLabel → {"npps=PointsPerCycle", "nppi=IntegrationPointsPerCycle", "Timing (s)"}
   ,
   PlotLabel → Row[{"time in seconds=", decoupledTimingsModel[100 "(npps/100)", 100 "(nppi/100)"]}]
   , ImageSize → 650
  ],
  ListPointPlot3D[
   (Flatten /@ Flatten[decoupledTimingsList, {{1, 2}}]) [[All, {1, 2, 3}]]
   , PlotStyle → {{Black, PointSize[Large]}}
  ]
 }]
```

time in seconds $=2.33613\,(\frac{\text{nppi}}{100})(\frac{\text{npps}}{100})+0.0260245\,(\frac{\text{nppi}}{100})-0.00404974\,(\frac{\text{npps}}{100})+0.0470921$



```
decoupledMemoryModel = LinearModelFit[
   (Flatten /@ Flatten[decoupledTimingsList, {{1, 2}}])[[All, {1, 2, 4}]]
   , {1, nSampling, nIntegration, nSampling × nIntegration}, {nSampling, nIntegration}];
Show[{
  Plot3D[
   decoupledMemoryModel[nSampling, nIntegration]
   , {nSampling, 0, 500}, {nIntegration, 0, 500}
   , AxesLabel → {"npps=PointsPerCycle", "nppi=IntegrationPointsPerCycle", "Memory (B)"}
   , PlotLabel → Row[{"Memory used (B)=", decoupledMemoryModel[100 "(npps/100)", 100 "(nppi/100)"]}]
   , ImageSize → 650
  ],
  ListPointPlot3D[
   (Flatten /@ Flatten[decoupledTimingsList, {{1, 2}}])[[All, {1, 2, 4}]]
   , PlotStyle → {{Black, PointSize[Large]}}
  ]
 }]
```

Memory used (B)=$4.72186\times10^6\left(\frac{\text{nppi}}{100}\right)\left(\frac{\text{npps}}{100}\right)+104696.\left(\frac{\text{nppi}}{100}\right)+65224.7\left(\frac{\text{npps}}{100}\right)-214229.$



Cutting off the long trajectories

This section shows, as an example of the use of the package, the use of the integration gate to eliminate the contribution from long trajectories. This can be tested by the reduced presence of quantum path interference patterns in the spectrum, and more practically by examining the dependence of the quantum phase on the field intensity.

The gating cutoff time

Given the classical trajectory,

```
trajectory[ωt_, ωt0_] := (x[ωt] /. First@DSolve[
      {x''[ωt] == Cos[ωt], x'[ωt0] == 0, x[ωt0] == 0}
      , x, ωt
     ])
```

the recollision kinetic energy and excursion time can be found as

```
recollisionKE[ωt0_?NumericQ] := (D[trajectory[ωtt, ωt0], ωtt]^2 /. ωtt → ωt) /.
   First[Quiet[NSolve[{trajectory[ωt, ωt0] == 0, π/2 ≤ ωt < 2 π}, ωt]]]
recollisionExcursionTime[ωt0_?NumericQ] :=
  (ωt - ωt0) /. First[Quiet[NSolve[{trajectory[ωt, ωt0] == 0, π/2 ≤ ωt < 2 π}, ωt]]]
```

and the excursion time at the cutoff can be found by maximizing the kinetic energy.

```
FindMaximum[recollisionKE[ωt0], {ωt0, 0.3}]
recollisionExcursionTime[ωt0]/(2 π) /. Last[%]
{1.58657, {ωt0 → 0.313408}}

0.650239
```

In other words, the cutoff trajectories occur at excursion times of $\omega\tau = 0.65\times 2\pi$, i.e. at a gate number of 0.65 cycles.

Calculation

This calculation runs a standard linearly-polarized field with intensity between 0.8 and $2\times10^{14}\,\mathrm{W}/\mathrm{cm}^2$, with a fine intensity resolution. We compare the standard, non-gated calculation against a calculation with nGate set to 0.65, as per the above, and a sharp $\sin^2$ cutoff of 0.05 cycles.

```
intRange = Range[0.8, 2., 0.002];
nppsl = 150;
SetSharedFunction[quantumPhaseScan, fourierDipole];
LaunchKernels[];
DistributeDefinitions["RBSFA`"];
nFlat = 0.65;
nGateRamp = 0.05;
```

The actual calculation,

```
DateString[]
AbsoluteTiming[
 ParallelTable[
   quantumPhaseScan[trajectories, int] = makeDipoleList[
      VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
      FieldParameters → {F → √int 0.053, ω → 0.057},
      PointsPerCycle → nppsl,
      If[trajectories === "Short", Sequence @@
         {Gate → SineSquaredGate[nGateRamp], nGate → (nFlat + nGateRamp)}, ## &[], ## &[]]
      ];
   , {int, intRange}, {trajectories, {"Short", "Long"}}];
]
DateString[]
```

```
Wed 2 Dec 2015 19:28:50

{1216.8, Null}

Wed 2 Dec 2015 19:49:07
```

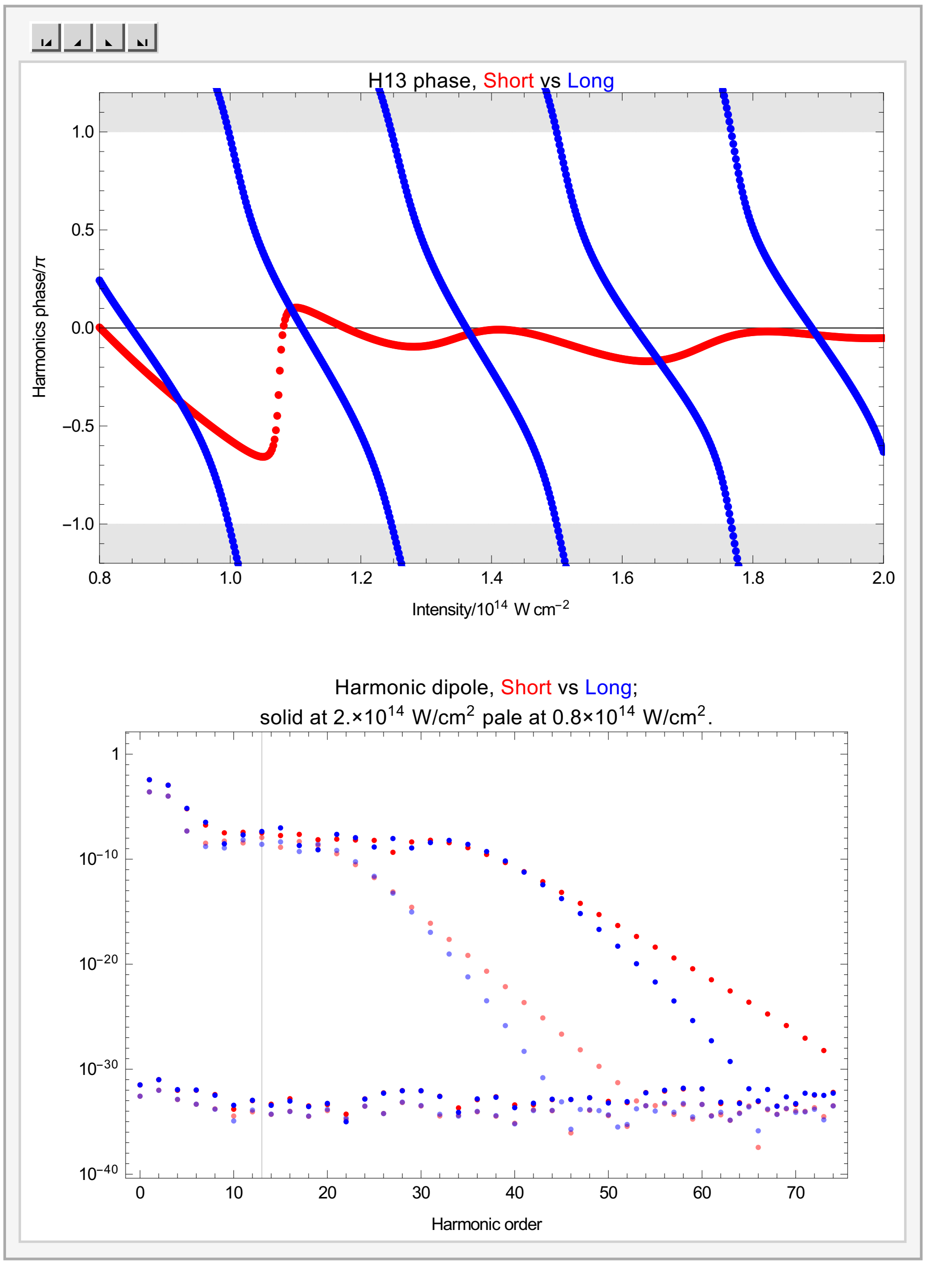
and the energy-domain dipole.

```
AbsoluteTiming[
 Table[
   fourierDipole[trajectories, int] = Fourier[
      Re[quantumPhaseScan[trajectories, int][[1 ;; -2, 1]]]
      , FourierParameters → {-1, 1}];
   , {int, intRange}, {trajectories, {"Short", "Long"}}];
]
```

```
{0.093762, Null}
```

Analysis

```
Block[{background},
 background = ListLogPlot[
   Flatten[Table[
     {Range[0, nppsl / 2 - 1], Abs[fourierDipole[trajectories, m[intRange]][[1 ;; nppsl / 2]]]^2}^T
     , {m, {Min, Max}}, {trajectories, {"Short", "Long"}}], 1]
   , ImageSize → 420
   , PlotStyle → {{Red, Opacity[0.5]}, {Blue, Opacity[0.5]}, {Red}, {Blue}}
   , PlotLabel → "Harmonic dipole, Short vs Long;\nsolid at " <> ToString[Max[intRange]] <>
     "×10^14 W/cm^2 pale at " <> ToString[Min[intRange]] <> "×10^14 W/cm^2."
   , FrameLabel → {"Harmonic order", ""}
   , Frame → True
  ];
 SlideView[Table[
   Row[{
     Show[{
       RegionPlot[Abs[ϕ] > 1, {int, Min[intRange], Max[intRange]}, {ϕ, -1.2, 1.2},
        PlotStyle → GrayLevel[0.9], Method → {"AxesInFront" → False}, BoundaryStyle → None],
       ListPlot[
        Table[
         Flatten[Table[
           {#, {#[[1]], #[[2]] + 2}, {#[[1]], #[[2]] - 2}} &@
            {int, 1/π Arg[fourierDipole[trajectories, int][[HO + 1]]]}
           , {int, intRange[[1 ;; -1]]}], 1]
          , {trajectories, {"Short", "Long"}}]
        , PlotStyle → {{PointSize[0.01], Red}, {PointSize[0.01], Blue}}
        , Joined → False
       ]
      }
      , PlotRange → 1.2 {-1, 1}, AspectRatio → 0.6
      , PlotRangePadding → {None, Automatic}
      , Axes → True
      , ImageSize → 450
      , PlotLabel → "H" <> ToString[HO] <> " phase, Short vs Long"
      , FrameLabel → {"Intensity/10^14 W cm^-2", "Harmonics phase/π"}
     ],
     Show[{background}, GridLines → {{HO}, None}]
    }]
   , {HO, 1, nppsl / 2, 2}], 7]
]
```



The short- and long-trajectory calculations are in red and blue respectively. It is clear that, in the plateau regions, the gated calculation has a much smoother dependence of the harmonic phase on the field intensity. On the other hand, the cutoff is perfectly preserved. These are the hallmarks that the contributions from long trajectories have been mostly eliminated.

Nondipole contributions

Nondipole contributions can be specified by setting a nonzero vector potential gradient:

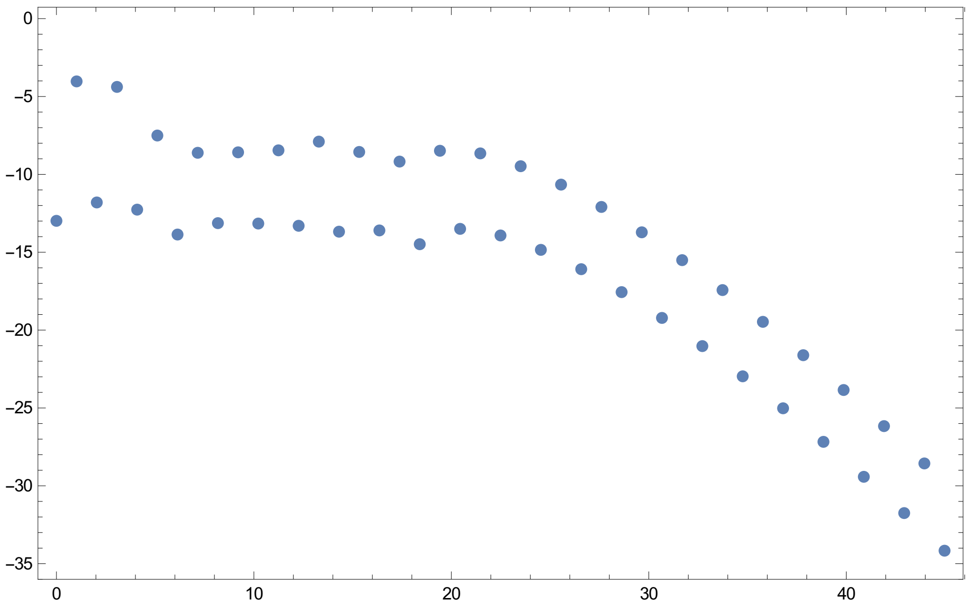
```
? VectorPotentialGradient
```

"VectorPotentialGradient is an option for makeDipole list which specifies the gradient of the field's vector potential. Usage should be VectorPotentialGradient→GA, where GA[t]//.pars must yield a square matrix of the same dimension as the vector potential for numeric t and parameters indicated by FieldParameters→pars. The indices must be such that GA[t]⟦i,j⟧ returns $\partial_i A_j[t]$."

If, for example, the travelling-wave form of the vector potential is of the form $\boldsymbol{A}(\boldsymbol{r},\,t) = \frac{F}{\omega}\hat{\boldsymbol{x}}\cos(k\,z - \omega\,t)$, then at the origin the vector potential is $\boldsymbol{A}(\mathbf{0}, t) = \frac{F}{\omega}\hat{\boldsymbol{x}}\cos(\omega t)$ and it has a single nonzero entry in its gradient matrix $\nabla\boldsymbol{A}$, i.e. $\partial_z A_x = -\frac{kF}{\omega}\sin(\omega t)$. This is entered into the VectorPotentialGradient option as

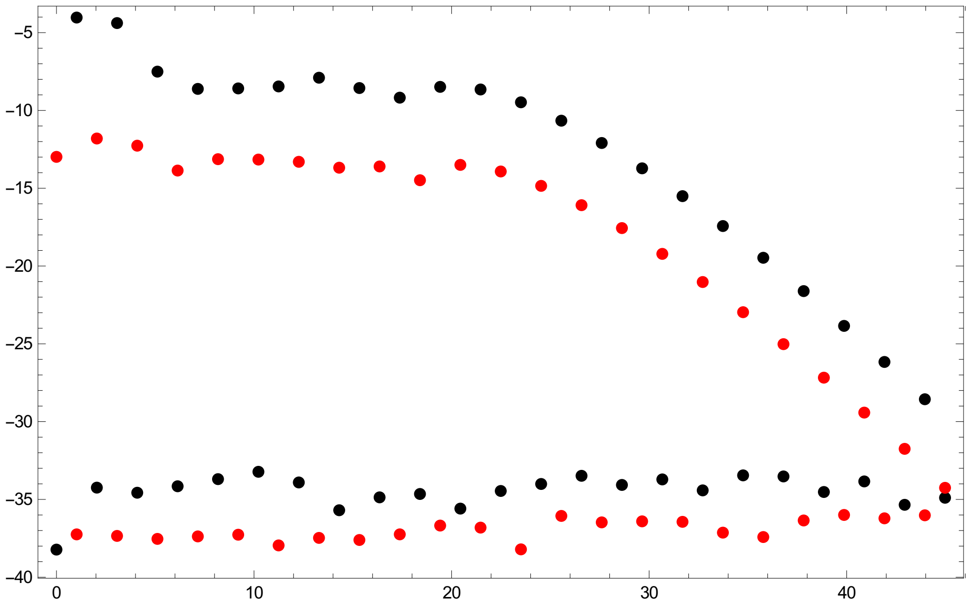
```
nonDipoleContributions = makeDipoleList[
    VectorPotential → Function[t, {F/ω Cos[ω t], 0, 0}],
    VectorPotentialGradient → Function[t, {{0, 0, 0}, {0, 0, 0}, {-k F/ω Sin[ω t], 0, 0}}],
    FieldParameters → {F → 0.05, ω → 0.057, k → ω / c, c → 137}
  ];
```

```
spectrumPlotter[getSpectrum[Most[nonDipoleContributions]], Joined → False, ImageSize → 500]
```



At low wavelengths, the first obvious effect is the appearance of even harmonics. This is the expected behaviour, with the harmonics along the laser propagation direction. (Informally, the magnetic pushing on the wavepacket acts on the propagation direction on both halves of each laser period. This off-axis recollision causes the dipole to oscillate in the propagation direction with an even symmetry. More formally, the dynamical symmetries of the problem permit even (but not odd) harmonics along this direction.) This is indeed what is observed:

```
Show[{
  spectrumPlotter[getSpectrum[nonDipoleContributions[[1 ;; -2, {1, 2}]]],
   Joined → False, PlotStyle → Black],
  spectrumPlotter[getSpectrum[nonDipoleContributions[[1 ;; -2, {3}]]],
   Joined → False, PlotStyle → Red]
 }, PlotRange → All, ImageSize → 500]
```



Benchmarking the nondipole contributions

Nondipole contributions in a crossed-beam setup

This section explores the harmonic emission by a crossed-beam setup, with nondipole contributions, as a benchmarking step for the latter. The crossed-beam setup was proposed by X.-M. Tong and S.-I. Chu in *Phys. Rev. A* **58** no .4, R2656 (1998), and it was explored in a nondipole setting by V. Averbukh et al. in *Phys Rev. A* **65,** 063402 (2002). The results below reproduce those of Averbukh et al.

In short, we consider the harmonic emission by a circularly polarized pulse propagating along the *z* direction, at frequency *ω*, and a linearly polarized pulse of frequency *r ω* propagating along the *x* direction and polarized along the *z* direction.

```
(crossedBeamsA[x_, z_] = Function[t,
    { F1             F1             F2
     {── Cos[k z - ω t], ── Sin[k z - ω t], ─── Sin[r k x - r ω t + θ0]}
      ω              ω             r ω
   ])[t] // MatrixForm
(crossedBeamsGA[x_] = Function[t, Evaluate[{
      D[crossedBeamsA[x, z][t], x] /. {z → 0},
      D[crossedBeamsA[x, z][t], y] /. {z → 0},
      D[crossedBeamsA[x, z][t], z] /. {z → 0}
     }]]
 )[t] // MatrixForm
```

$$\begin{pmatrix} \frac{F1\,Cos[k\,z-t\,\omega]}{\omega} \\ \frac{F1\,Sin[k\,z-t\,\omega]}{\omega} \\ \frac{F2\,Sin[k\,r\,x+\theta0-r\,t\,\omega]}{r\,\omega} \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & \frac{F2\,k\,Cos[k\,r\,x+\theta0-r\,t\,\omega]}{\omega} \\ 0 & 0 & 0 \\ \frac{F1\,k\,Sin[t\,\omega]}{\omega} & \frac{F1\,k\,Cos[t\,\omega]}{\omega} & 0 \end{pmatrix}$$

The dipole selection rules allow harmonic orders of the form $2rl \pm 1$, with $l = 0, 1, 2, 3,$ ..., with polarization in the *x*, *y* plane, and harmonics of order $r(2l+1)$, with $l = 0, 1, 2, 3,$ ..., polarized along the *z* direction.

```
allowedHarmonics[r_, {1, 2}] := Select[Union[2 r Range[0, 100] + 1, 2 r Range[0, 100] - 1], # > 0 &]
allowedHarmonics[r_, {3}] := r (2 Range[0, 100] + 1)
```

For the calculation, then, some preliminaries,

```
αRange = {0, 1 / 137};
nppcb = 240;
crossedBeamsParameters[rr_] := {F1 → 0.1, F2 → 0.2, ω → 0.057, θ0 → 0, r → rr, k → α ω};
DistributeDefinitions["RBSFA`"];
SetSharedFunction[crossedBeamsResults];
```

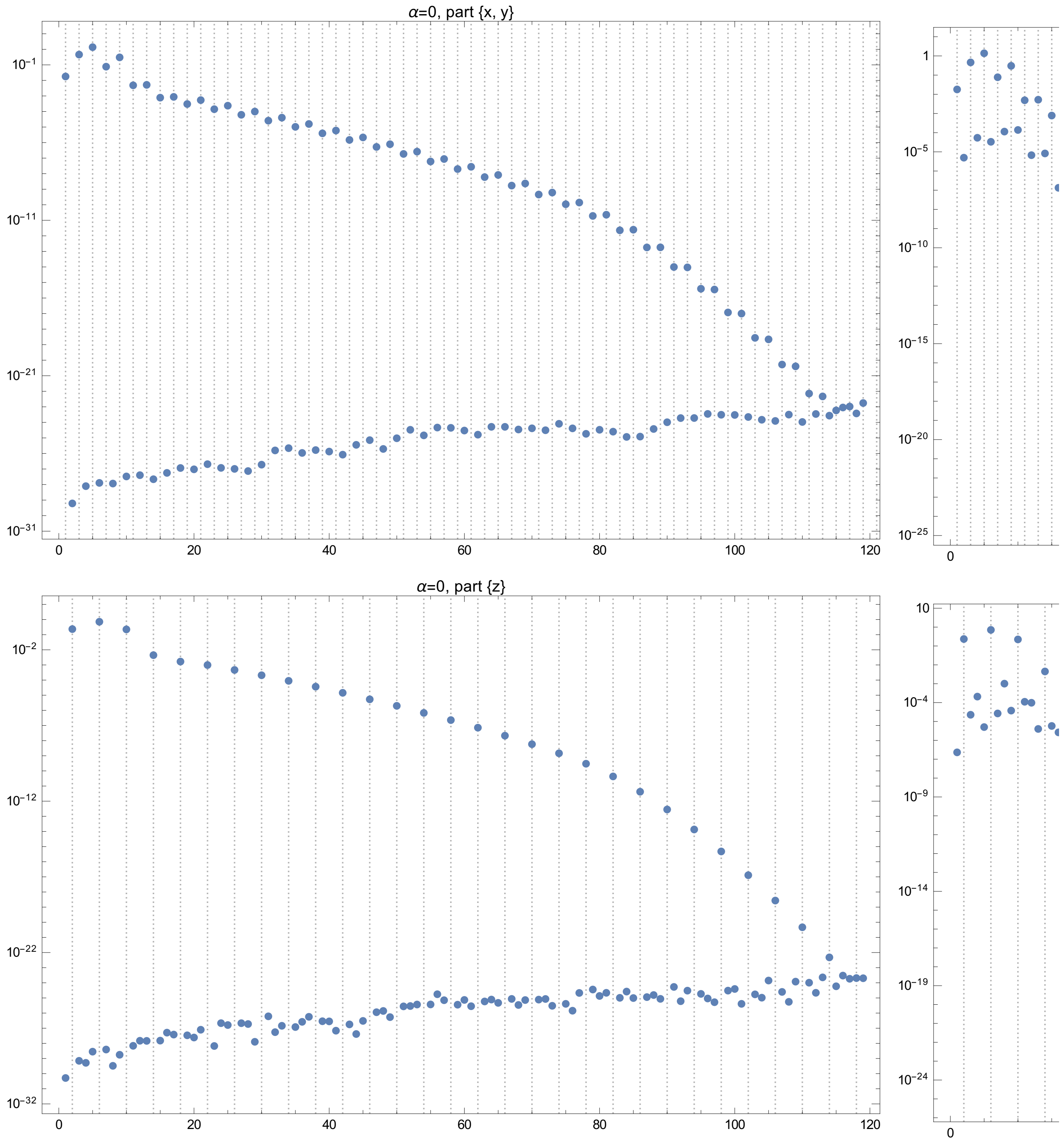
and the calculation itself for $r = 2$ and $r = 5$.

```
Print[DateString[]]
ParallelTable[AbsoluteTiming[
  crossedBeamsResults[r, α] = makeDipoleList[
     VectorPotential → crossedBeamsA[0, 0], VectorPotentialGradient → crossedBeamsGA[0],
     FieldParameters → crossedBeamsParameters[r]
     , DipoleTransitionMatrixElement → Function[{p, κ}, gaussianDTME[p, 1 / 1.3]]
     , CarrierFrequency → 0.057, PointsPerCycle → nppcb
    ];
 ], {r, {2, 5}}, {α, αRange}]
Print[DateString[]]
```

```
Thu 1 Oct 2015 15:42:15

{{{29.6518, Null}, {34.7323, Null}}, {{28.9947, Null}, {34.9678, Null}}}

Thu 1 Oct 2015 15:43:20
```

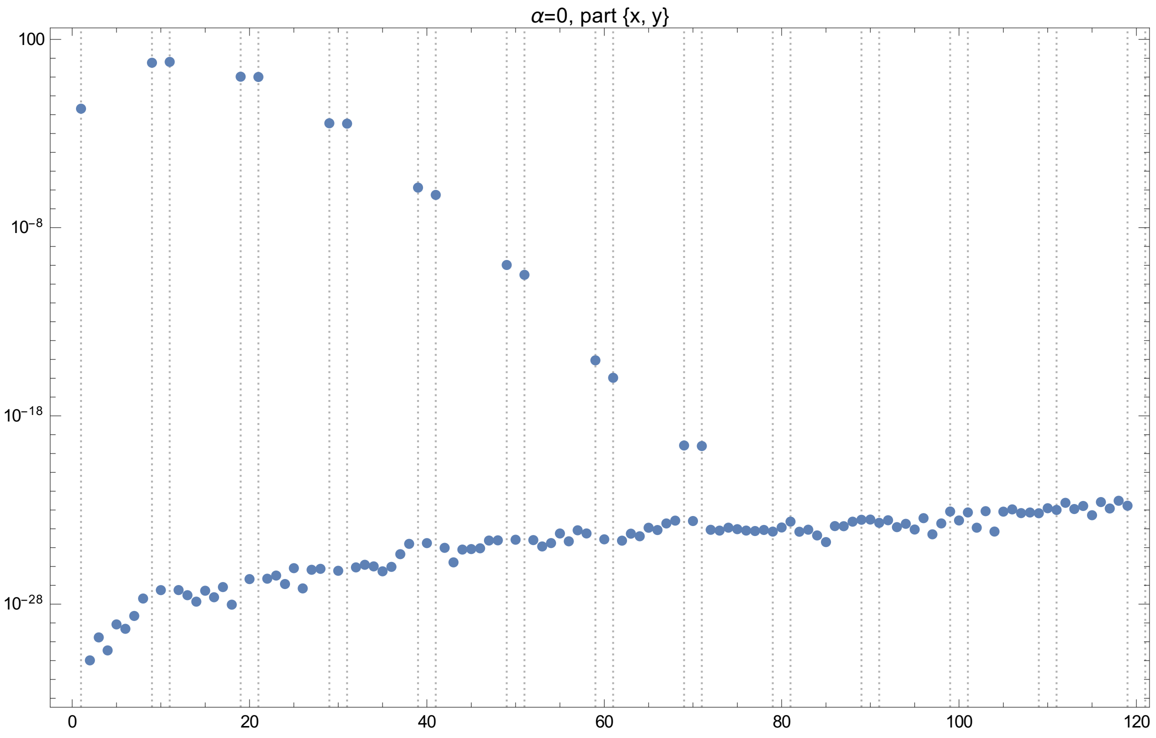
Results for $r = 2$, comparable to Fig. 1 in Averbukh et al. Dashed lines mark the dipole-allowed harmonics. The left-hand column has nondipole contributions turned off ($\alpha = 0$), and the right-hand column includes the nondipole contributions and observes a massive increase in the amplitude of the dipole-forbidden harmonics.

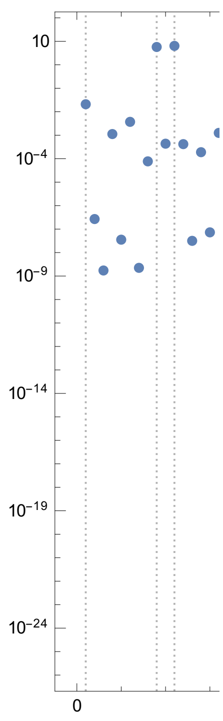
```
Grid[Table[
  ListLogPlot[
   getSpectrum[crossedBeamsResults[2, α][[1 ;; -2, part]], ωPower → 2][[2 ;;]]
   , Joined → False, ImageSize → 600, PlotTheme → "Detailed", PlotRange → Full
   , GridLines → {allowedHarmonics[2, part], None}
   , PlotLabel → Row[{"α=", α, ", part ", {"x", "y", "z"}[[part]]}]
  ], {part, {{1, 2}, {3}}}, {α, αRange}]]
```

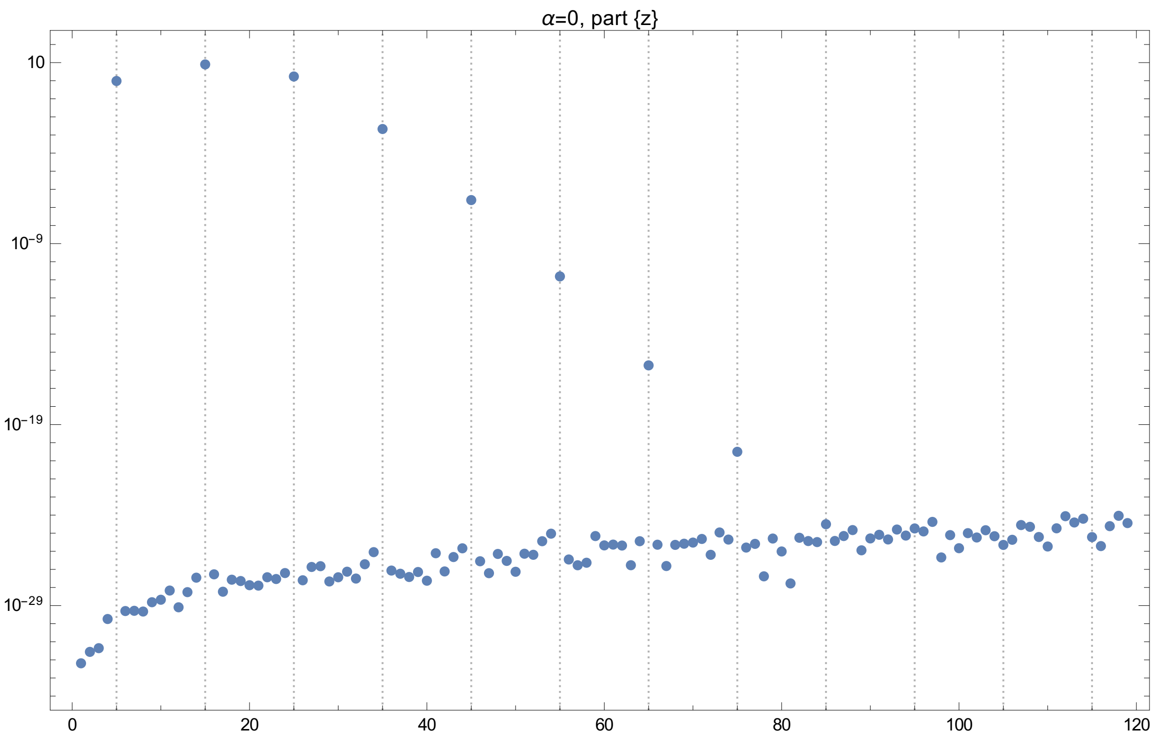


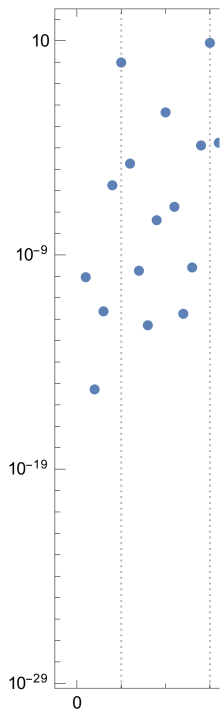
Results for $r = 5$, comparable to Fig. 2 in Averbukh et al.

```
Grid[Table[
  ListLogPlot[
   getSpectrum[crossedBeamsResults[5, α]⟦1 ;; -2, part⟧, ωPower → 2]⟦2 ;;⟧
    , Joined → False, ImageSize → 600, PlotTheme → "Detailed", PlotRange → Full
    , GridLines → {allowedHarmonics[5, part], None}
    , PlotLabel → Row[{"α=", α, ", part ", {"x", "y", "z"}⟦part⟧}]
  ], {part, {{1, 2}, {3}}}, {α, αRange}]]
```









Multiple plateaus in HHG in ions

This section benchmarks this code against the results of N.J. Kylstra et al. reported in *J. Phys B: At. Mol. Opt. Phys.* **34** no. 3, L55 (2001);. In particular, we study HHG in the He$^+$ ion at high intensity ($I = 5.6\times10^{15}\,\mathrm{W\,cm^{-2}}$) and reasonable (800nm) wavelength.

```
(kylstraA[z_] = Function[t, {F/ω Sin[(ω t - k z)/4]^2 Sin[ω t - k z], 0, 0}])[t] // MatrixForm
(kylstraGA = Function[t, Evaluate[{
        D[kylstraA[z][t], x] /. {z → 0},
        D[kylstraA[z][t], y] /. {z → 0},
        D[kylstraA[z][t], z] /. {z → 0}
       }]]
  )[t] // MatrixForm
```

$$\begin{pmatrix} -\frac{F\,\mathrm{Sin}[k\,z-t\,\omega]\,\mathrm{Sin}\left[\frac{1}{4}(-k\,z+t\,\omega)\right]^2}{\omega} \\ 0 \\ 0 \end{pmatrix}$$

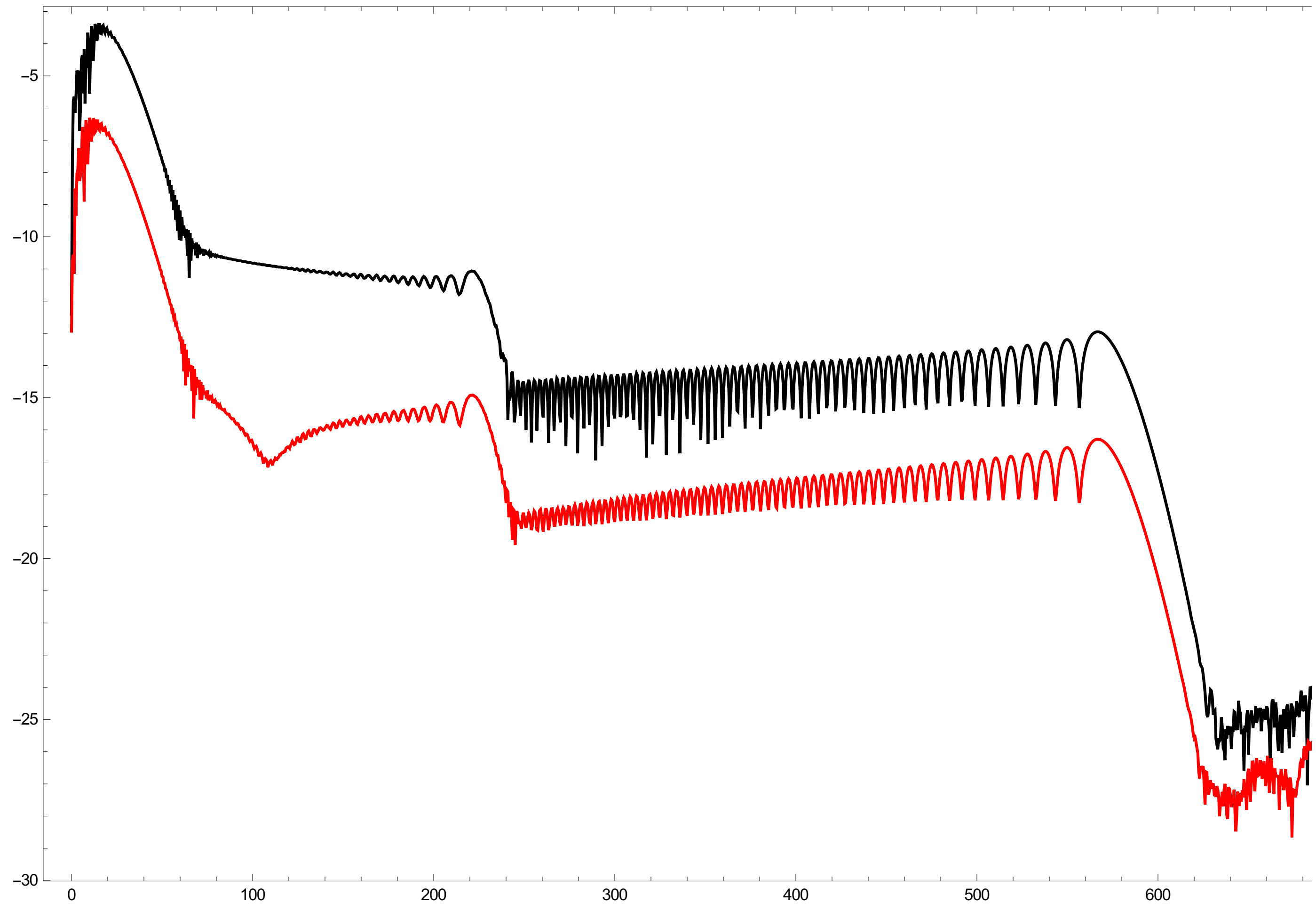
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -\frac{F\,k\,\mathrm{Cos}[t\,\omega]\,\mathrm{Sin}\left[\frac{t\,\omega}{4}\right]^2}{\omega}-\frac{F\,k\,\mathrm{Cos}\left[\frac{t\,\omega}{4}\right]\,\mathrm{Sin}\left[\frac{t\,\omega}{4}\right]\,\mathrm{Sin}[t\,\omega]}{2\,\omega} & 0 & 0 \end{pmatrix}$$

```
nppk = 1500;

DateString[]
AbsoluteTiming[
 kylstraTest = makeDipoleList[
     VectorPotential → kylstraA[0], VectorPotentialGradient → kylstraGA,
     FieldParameters → {F → √(5.6 × 10^15 / 10^14) 0.053, ω → 0.057, k → α ω, α → 1 / 137},
     IonizationPotential → 2,
     PointsPerCycle → nppk, TotalCycles → 2
    ];
]
DateString[]
```

```
Thu 1 Oct 2015 18:25:47

{1619.16, Null}

Thu 1 Oct 2015 18:52:46
```

Plotting the results. The *x* component (along the laser polarization) is in black, the *z* component (along the laser propagation) is in red.

```
Show[
 Table[
  spectrumPlotter[getSpectrum[kylstraTest⟦1 ;; -2, part⟧, ωPower → 2], Joined → True,
   PointsPerCycle → nppk, TotalCycles → 2, PlotStyle → (part /. {{1, 2} → Black, {3} → Red})]
  , {part, {{1}, {3}}}]
 , PlotRange → All
]
```



The results are a good qualitative match to the dipoles reported by Kylstra et al., with the notable exception of the low-order harmonics below n≲50.

On the other hand, taken naively this code cannot be applied to the harder targets described in that paper (Li^{2+} and Be^{3+}, at intensities between 0.9 and 3.6×10^{17} W cm^{-2}), which have cutoffs of order as high as 35 000, which requires several days to several months of calculation using the naive scaling. (That said, using a smarter choice of ReportingFunction, judicious use of parallelization and lots of waiting, those targets are probably within reach of this code.)

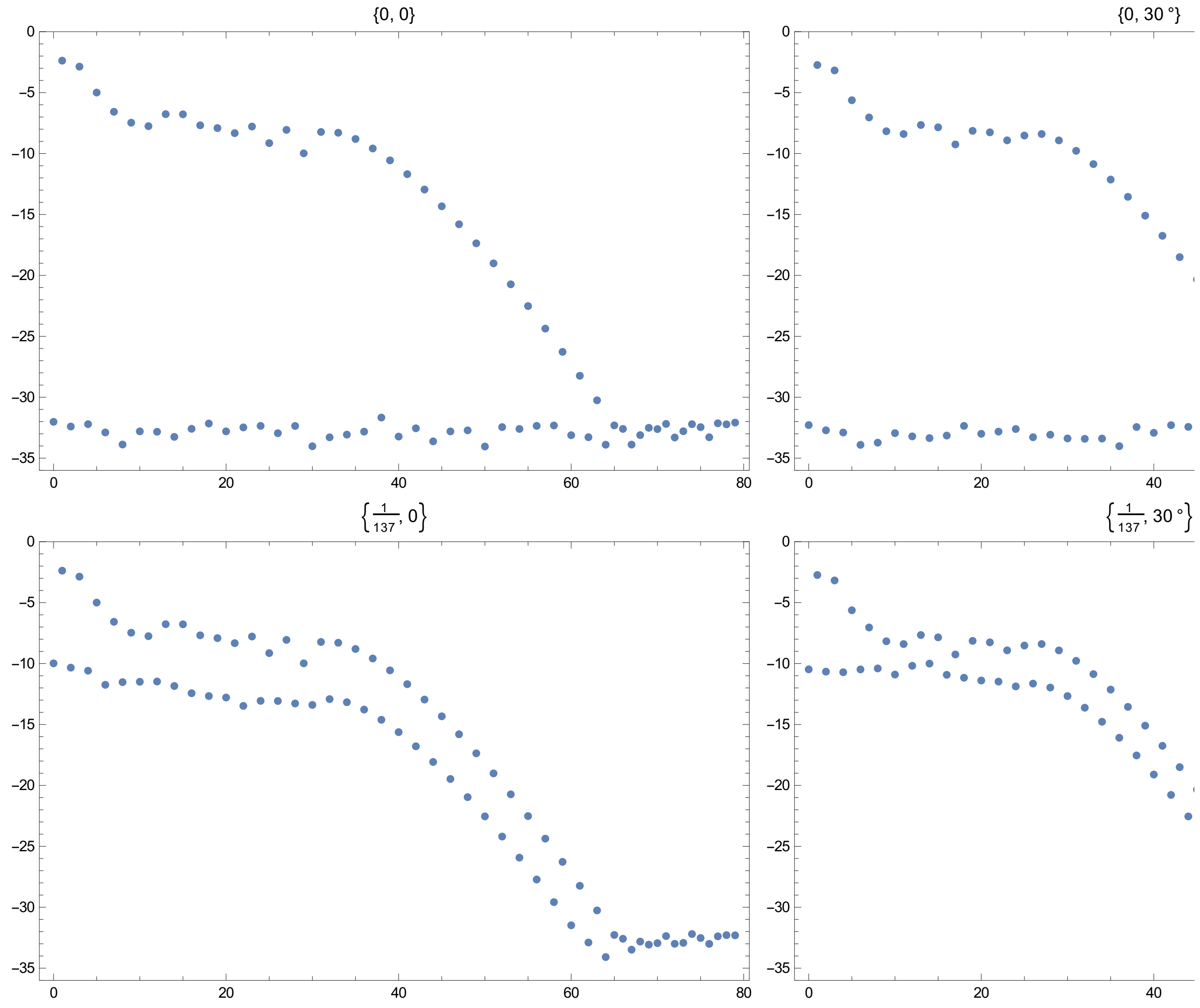
Perodicity of nondipole contributions

```
Quit
```

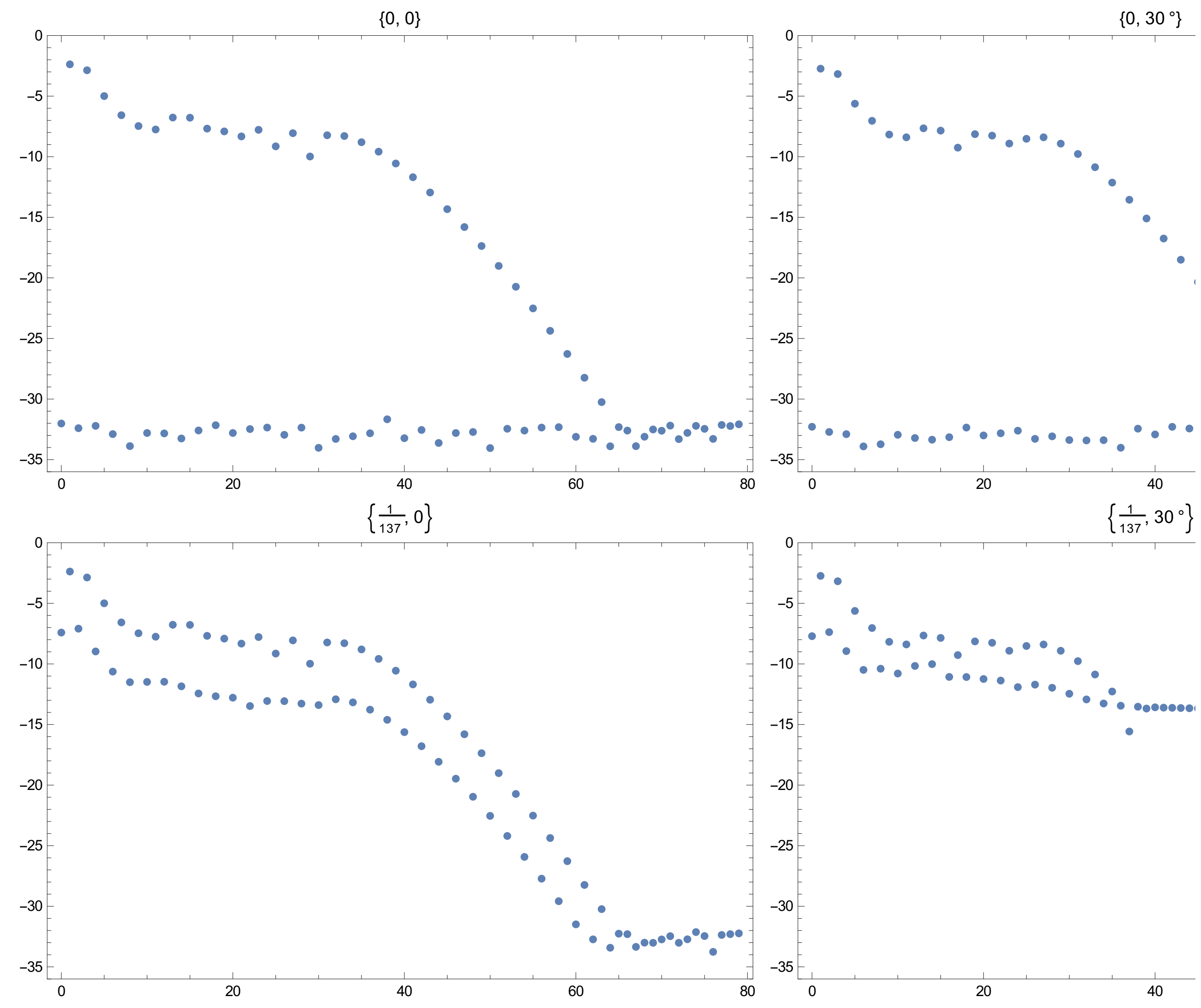
```
(crossedBeamsA[t_] = First /@ Sum[
        F/ω ( Cos[θ] ; 0 ; -s Sin[θ] ) Cos[k {s Sin[θ], 0, Cos[θ]}.{x, y, z} - ω t - s ϕ]
        , {s, {-1, 1}}]) // MatrixForm;
(crossedBeamsGA[t_] = Evaluate[{
       D[crossedBeamsA[t], x],
       D[crossedBeamsA[t], y],
       D[crossedBeamsA[t], z]
      }]) // MatrixForm;
crossedBeamsParameters = {x → 0, y → 0, z → 0, F → √0.5 0.053, ω → 0.057, θ → 10. °, k → α ω};

DateString[]
Table[
 First[AbsoluteTiming[
   symmetryTestDipole[α, ϕ] = makeDipoleList[
      VectorPotential → crossedBeamsA,
      VectorPotentialGradient → crossedBeamsGA, FieldParameters → crossedBeamsParameters,
      PointsPerCycle → 160
     ];
  ]]
 , {α, {0, 1 / 137}}, {ϕ, {0, 30 °}}]
DateString[]
```

```
Fri 16 Oct 2015 12:09:53

{{5.97834, 6.65576}, {8.31512, 11.9895}}

Fri 16 Oct 2015 12:10:26
```

```
Grid[Table[
  spectrumPlotter[
   getSpectrum[symmetryTestDipole[α, ϕ]〚1 ;; -2, {1, 2, 3}〛]
   , Joined → False, PointsPerCycle → 160,
   ImageSize → 450, PlotLabel → {α, ϕ}, PlotRange → {-36, 0}
  ]
  , {α, {0, 1 / 137}}, {ϕ, {0, 30 °}}]]
```



Compare this with the unacceptably high noise floor from the previous version for the case $\alpha = 1/137$, $\phi = 30°$.



Debugging and benchmarking tools

If something goes funny with your calls, then before you start taking makeDipoleList apart you can try using its Verbose option to diagnose the internal functions it is using. In particular:

- Setting Verbose→1 makes makeDipoleList print the Information of the key internal functions it is using, before it goes on to the integration loop.

```
makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
  FieldParameters → {F → 0.05, ω → 0.057}, Verbose → 1]⟦1 ;; 10⟧
```

```
RBSFA`Private`A
```

```
RBSFA`Private`A[RBSFA`Private`t$_] = {0.877193 Sin[0.057 RBSFA`Private`t$], 0, 0}
```

```
RBSFA`Private`GA
```

```
RBSFA`Private`GA[RBSFA`Private`t$_] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}
```

```
RBSFA`Private`ps
```

```
RBSFA`Private`ps[RBSFA`Private`t_?RBSFA`Private`gridPointQ,
  RBSFA`Private`tt_?RBSFA`Private`gridPointQ] :=
 RBSFA`Private`ps[RBSFA`Private`t, RBSFA`Private`tt] =
  - (Inverse[IdentityMatrix[Length[RBSFA`Private`A[RBSFA`Private`tInit]]] -
        (RBSFA`Private`GAIntInt[RBSFA`Private`t, RBSFA`Private`tt] +
           Transpose[RBSFA`Private`GAIntInt[RBSFA`Private`t, RBSFA`Private`tt]]) /
         (RBSFA`Private`t - RBSFA`Private`tt - ⅈ RBSFA`Private`ϵ)].
      (RBSFA`Private`AInt[RBSFA`Private`t, RBSFA`Private`tt] -
         RBSFA`Private`bigPScorrectionInt[RBSFA`Private`t, RBSFA`Private`tt]) /
     (RBSFA`Private`t - RBSFA`Private`tt - ⅈ RBSFA`Private`ϵ))
```

```
RBSFA`Private`pi
```

```
RBSFA`Private`pi[RBSFA`Private`p_, RBSFA`Private`t_] :=
 RBSFA`Private`p + RBSFA`Private`A[RBSFA`Private`t] -
  RBSFA`Private`GAInt[RBSFA`Private`t].RBSFA`Private`p - RBSFA`Private`GAdotAInt[RBSFA`Private`t]
```

```
RBSFA`Private`S
```

```
RBSFA`Private`S[RBSFA`Private`t_?RBSFA`Private`gridPointQ,
  RBSFA`Private`tt_?RBSFA`Private`gridPointQ] :=
 1/2 (Norm[RBSFA`Private`ps[RBSFA`Private`t, RBSFA`Private`tt]]^2 + RBSFA`Private`κ^2)
   (RBSFA`Private`t - RBSFA`Private`tt) + RBSFA`Private`ps[RBSFA`Private`t, RBSFA`Private`tt].
   RBSFA`Private`AInt[RBSFA`Private`t, RBSFA`Private`tt] +
  1/2 RBSFA`Private`A2Int[RBSFA`Private`t, RBSFA`Private`tt] -
  (RBSFA`Private`ps[RBSFA`Private`t, RBSFA`Private`tt].RBSFA`Private`GAIntInt[
      RBSFA`Private`t, RBSFA`Private`tt].RBSFA`Private`ps[RBSFA`Private`t, RBSFA`Private`tt] +
    RBSFA`Private`ps[RBSFA`Private`t, RBSFA`Private`tt].
     RBSFA`Private`bigPScorrectionInt[RBSFA`Private`t, RBSFA`Private`tt] +
    RBSFA`Private`AdotGAdotAInt[RBSFA`Private`t, RBSFA`Private`tt])
```

```
RBSFA`Private`AInt
```

```
RBSFA`Private`AInt[RBSFA`Private`t$_] = {-15.3894 Cos[0.057 RBSFA`Private`t$], 0, 0}

RBSFA`Private`AInt[RBSFA`Private`t$_, RBSFA`Private`tt$_] =
 {15.3894 Cos[0.057 RBSFA`Private`tt$] - 15.3894 Cos[0.057 RBSFA`Private`t$], 0, 0}
```

```
RBSFA`Private`A2Int
```

```
RBSFA`Private`A2Int[RBSFA`Private`t$_] =
 0.769468 (0.5 RBSFA`Private`t$ - 4.38596 Sin[0.114 RBSFA`Private`t$])

RBSFA`Private`A2Int[RBSFA`Private`t$_, RBSFA`Private`tt$_] =
 - 0.769468 (0.5 RBSFA`Private`tt$ - 4.38596 Sin[0.114 RBSFA`Private`tt$]) +
  0.769468 (0.5 RBSFA`Private`t$ - 4.38596 Sin[0.114 RBSFA`Private`t$])
```

```
RBSFA`Private`GAInt
```

```
RBSFA`Private`GAInt[RBSFA`Private`t$_] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}

RBSFA`Private`GAInt[RBSFA`Private`t$_, RBSFA`Private`tt$_] = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}
```

(abridged.)

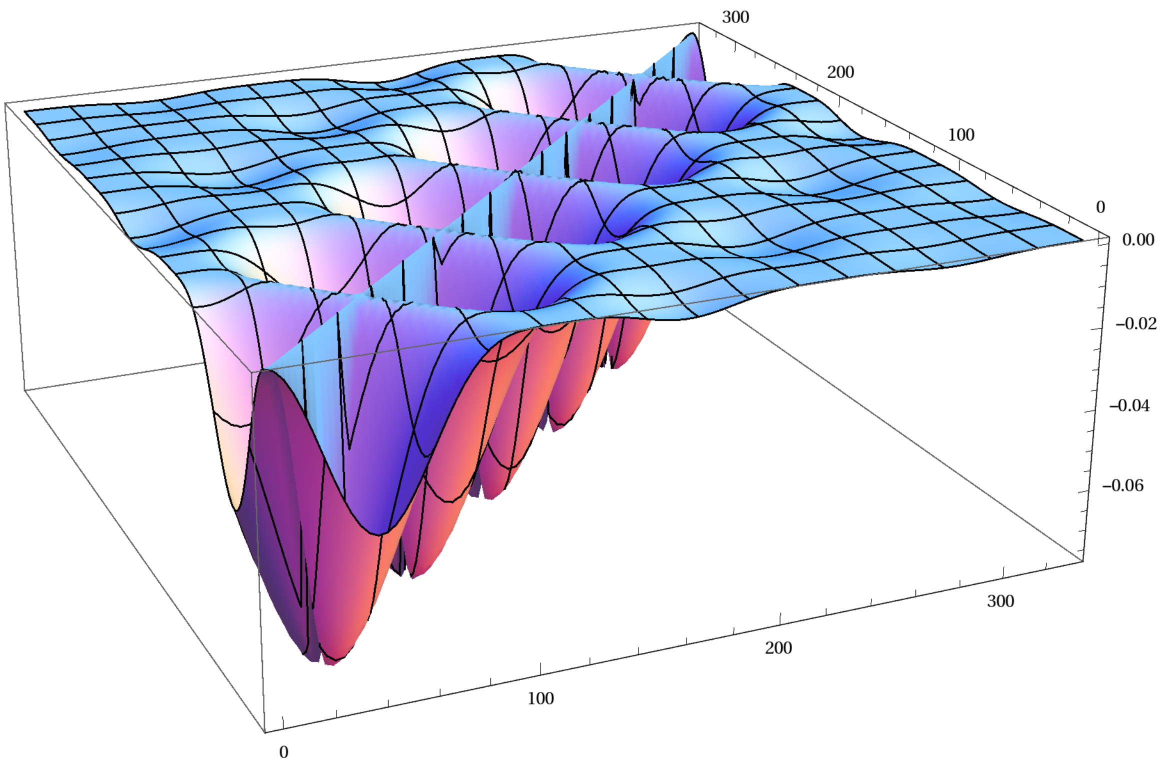
```
{{-0.0198736 + 0.00113629 i, 0. + 0. i, 0. + 0. i}, {-0.0166186 - 1.44349 i, 0. + 0. i, 0. + 0. i},
 {-0.0132498 - 5.34015 i, 0. + 0. i, 0. + 0. i}, {-0.00957477 - 10.5721 i, 0. + 0. i, 0. + 0. i},
 {-0.00563402 - 15.8027 i, 0. + 0. i, 0. + 0. i}, {-0.00185819 - 19.9418 i, 0. + 0. i, 0. + 0. i},
 {0.00123386 - 22.4066 i, 0. + 0. i, 0. + 0. i}, {0.00353492 - 23.1295 i, 0. + 0. i, 0. + 0. i},
 {0.0053967 - 22.4 i, 0. + 0. i, 0. + 0. i}, {0.00707192 - 20.6646 i, 0. + 0. i, 0. + 0. i}}
```

- Setting Verbose→2 makes makeDipoleList output its key internal functions and shut down before the integration takes place. Its results can be caught as follows:

```
{A[t_], GA[t_], ps[t_, tt_], pi[p_, t_], S[t_, tt_], AInt[t_], AInt[t_, tt_], A2Int[t_],
   A2Int[t_, tt_], GAInt[t_], GAInt[t_, tt_], GAdotAInt[t_], GAdotAInt[t_, tt_], AdotGAInt[t_],
   AdotGAInt[t_, tt_], GAIntInt[t_], GAIntInt[t_, tt_], bigPScorrectionInt[t_],
   bigPScorrectionInt[t_, tt_], AdotGAdotAInt[t_], AdotGAdotAInt[t_, tt_], integrand[t_, τ_]
  } = makeDipoleList[VectorPotential → Function[t, {F/ω Sin[ω t], 0, 0}],
   FieldParameters → {F → 0.05, ω → 0.057}, Verbose → 2];
```

This then enables examination of e.g. the action:

```
Block[{ω = 0.057},
 Plot3D[
  Im[S[t, tt]]
  , {t, 0, 3 2π/ω}, {tt, 0, 3 2π/ω}
  , PlotRange → Full, ImageSize → 600, PlotTheme → "Classic", PlotPoints → 100
 ]
]
```



See the implementation notes in the code for makeDipoleList for further definitions of what each term entails.

Bicircular fields

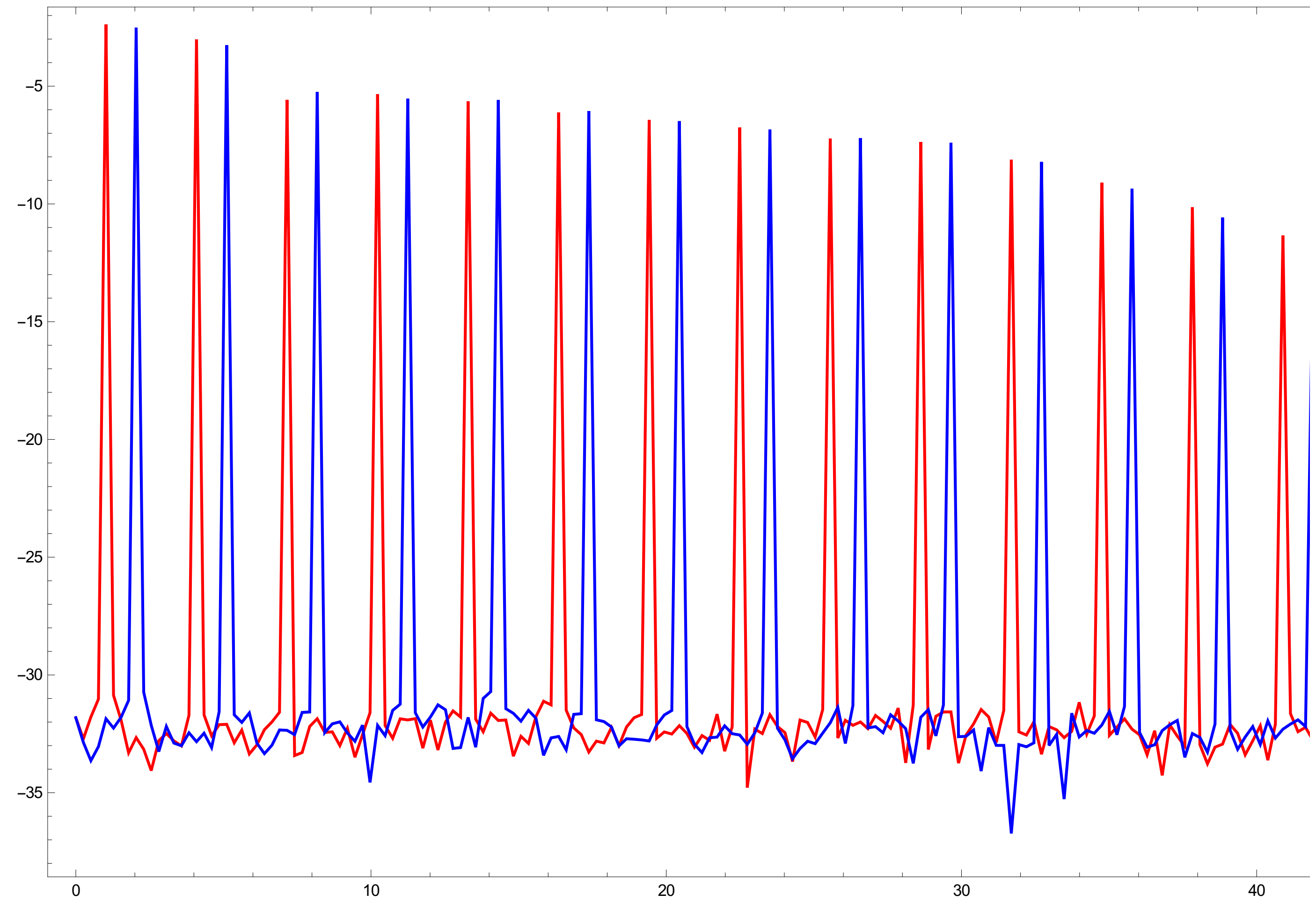
As a slightly less trivial example, consider a bicircular field: two counter-rotating, circularly polarized fields of different frequencies. The 'standard' case - as first demonstrated experimentally - has one field as the second harmonic of the fundamental, with both at equal intensities. The resultant harmonics appear at all integer orders except those divisible by three, with the $3n+1$ harmonics polarized as the fundamental, and the $3n-1$ harmonics polarized as the second-harmonic driver.

```
bicircularA[t_] :=
 (F1/ω1 {Cos[t ω1] Sin[α], - Cos[α] Sin[t ω1]} + F2/ω2 {Cos[β] Cos[ω2 t], Sin[β] Sin[ω2 t]})
bicircularParameters = {F1 → 0.075, F2 → 0.075, α → 45 °, β → 45 °, ω1 → 45.6 / 800, ω2 → 45.6 / 400};
AbsoluteTiming[bicircularTest = makeDipoleList[VectorPotential → bicircularA,
    FieldParameters → bicircularParameters, TotalCycles → 4];]
```

```
{8.45739, Null}
```

The function biColorSpectrum takes the spectrum and plots it, separating the two circular polarizations into different colours.

```
biColorSpectrum[Most[bicircularTest]]
```



Bicircular fields with a sine-squared envelope

To benchmark the original calculations, we compared them with the output of full MCTDH calculations. Here we used a $\sin^2$ envelope as the TDSE numerics require a finite pulse; the calculations take correspondingly longer but they are still very manageable (two/three minutes per calculation for a fifteen-cycle pulse, resolving up to ~70 harmonics). One distinctive feature is that the harmonics near the cutoff are broader, because less cycles contribute to those energies.

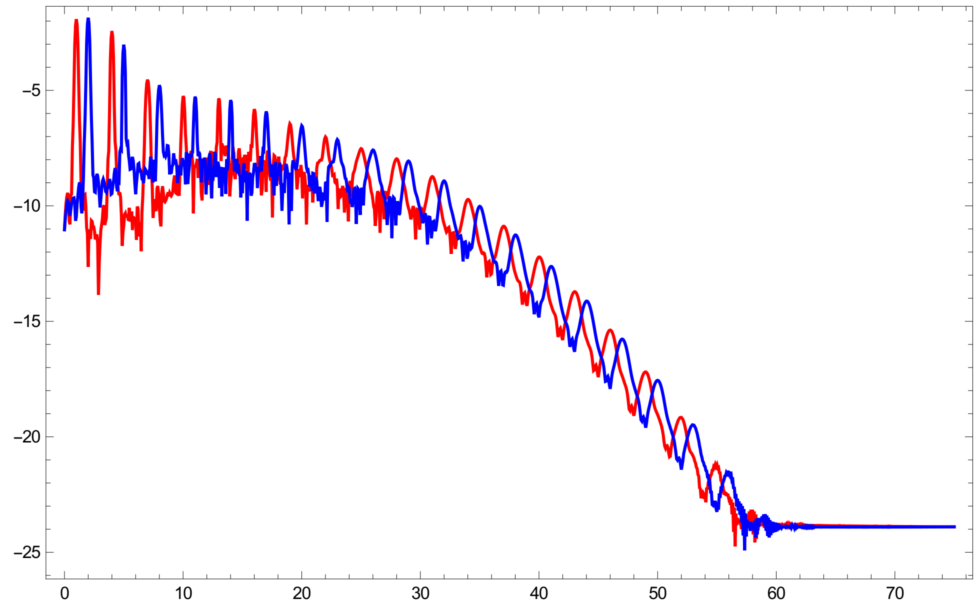
```
bicircularEnvelopeA[t_] := cosPowerFlatTop[ω1, TotalCycles, 2][t]
    (F1/ω1 {Cos[t ω1] Sin[α], - Cos[α] Sin[t ω1]} + F2/ω2 {Cos[β] Cos[ω2 t], Sin[β] Sin[ω2 t]});
bicircularParameters = {F1 → 0.075, F2 → 0.075, α → 45 °, β → 45 °, ω1 → 45.6 / 800, ω2 → 45.6 / 400};
```

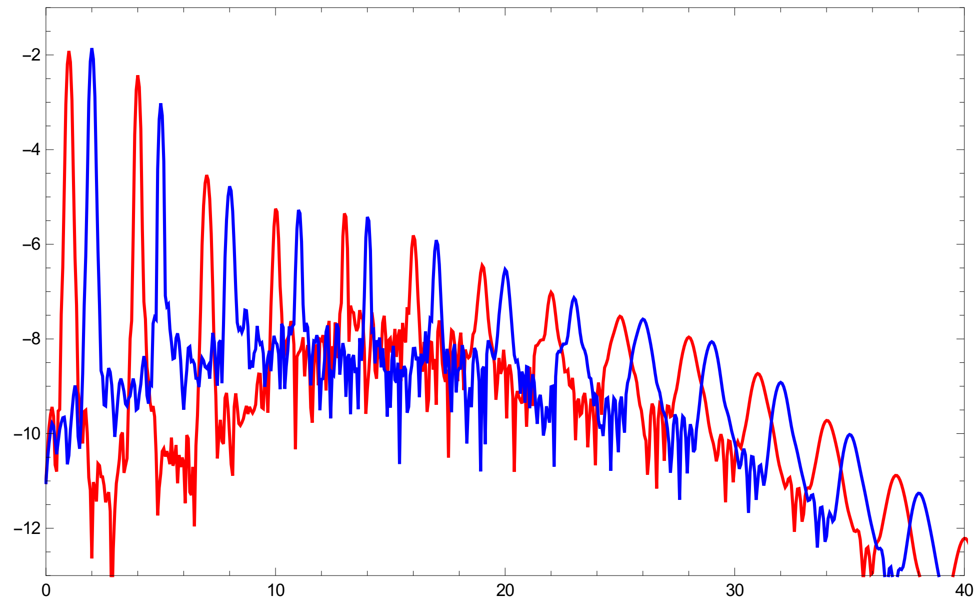
If (as in this case) the field depends on a number-of-cycles parameter, care must be taken that it matches the num option of the main call.

```
AbsoluteTiming[
 bicircularEnvelopeDipole =
   makeDipoleList[VectorPotential → bicircularEnvelopeA, FieldParameters →
     Join[bicircularParameters, {TotalCycles → 15}], PointsPerCycle → 150, TotalCycles → 15];
]
{141.302, Null}
```

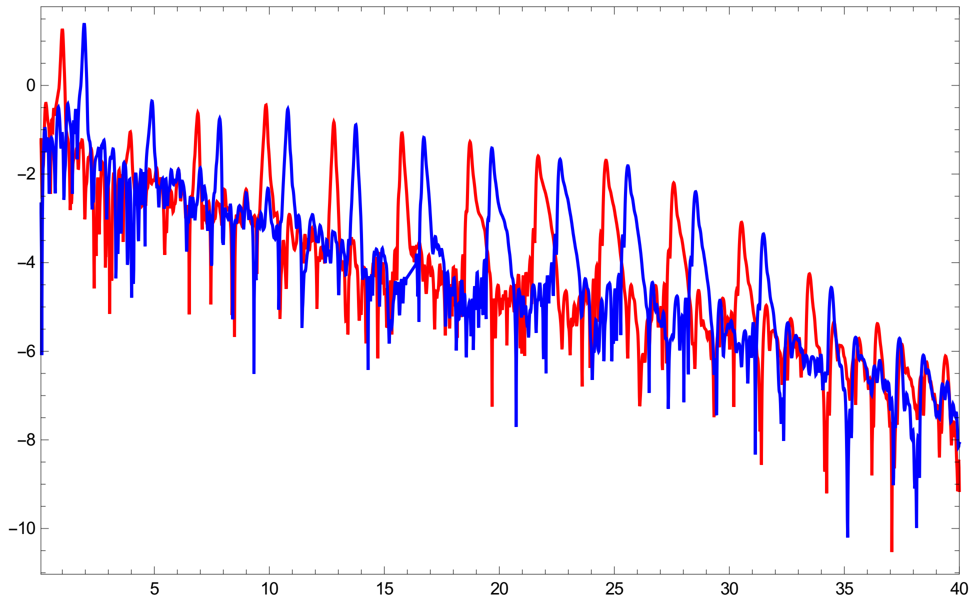
Plotting the spectrum, and a zoom at the plateau:

```
biColorSpectrum[bicircularEnvelopeDipole,
 PointsPerCycle → 150, TotalCycles → 15, ImageSize → 500]
biColorSpectrum[bicircularEnvelopeDipole, PointsPerCycle → 150,
 TotalCycles → 15, ImageSize → 500, PlotRange → {{0, 40}, {-13, -1}}]
```





The comparable MCTDH spectrum, for identical conditions, looks like this:



Original RB-SFA: ‘rotating’ bicircular fields

Calculation

Here the fundamental laser driver has been set at an elliptical polarization (as in the original experiment, A. Fleischer et al., *Nature Photon.* **8**, 543 (2014)), which helps investigate the spin-angular-momentum conservation properties of HHG. In the model proposed in the original paper (*Phys. Rev. A* **90**, 043829 (2014)), the photon model is validated by splitting the elliptical field itself into two circular components, which can then be tuned independently:

```
rotatingBicircularA[t_] := envelope[t] (F2/ω2 {Cos[β] Cos[ω2 t - ϕ1], Sin[β] Sin[ω2 t - ϕ1]} +
      F1/√2 (1/ω1 Cos[α - π/4] {Cos[ω1 t + ϕ1], -Sin[ω1 t + ϕ1]} +
          1/((1 + δ) ω1) Sin[α - π/4] {Cos[(1 + δ) ω1 t - ϕ1 + ϕ2], +Sin[(1 + δ) ω1 t - ϕ1 + ϕ2]}));
DistributeDefinitions["RBSFA`"];
directory = FileNameJoin[{NotebookDirectory[], "Temp Data"}];
filename[δ_] :=
  FileNameJoin[{directory, "data 25.09 detuning scan at δ=" <> ToString[δ] <> ".txt"}];
Length[δRange = Range[0, 0.25, 0.001]]
```

```
251
```

To test the validity of the photon model, we ran a scan over the detuning *δ*, using the calculation below.

```
DateString[]
Print["Total = ", Length[δRange], " points at ~230s/point will be done at approximately ",
 DateString[AbsoluteTime[] + Length[δRange] * 230. / 7], "."]
ParallelTable[
  Print[AbsoluteTiming[
    makeDipoleList[
      VectorPotential → rotatingBicircularA,
      FieldParameters → {α → 35 °, β → 45 °, F1 → 0.075, F2 → 0.075, ω1 → 0.057,
        ω2 → 1.95 × 0.057, ϕ1 → 0, ϕ2 → 0, envelope → flatTopEnvelope[ω1, 26, 3]},
      CarrierFrequency → 0.057, TotalCycles → 26, PointsPerCycle → 115,
      nGate → 1.8, PointNumberCorrection → 1, Preintegrals → "Numeric",
      ReportingFunction → Function[Write[filename[δ], #]]
     ];]];
  Print[DateString[]];
  , {δ, δRange}];
DateString[]
NotebookSave[]
```

Total time 2h 32min. (Desktop machine with 8-thread, 4-core Intel i7-3770 CPU at 3.40GHz, 16GB RAB, running 7 *Mathematica* kernels in parallel.)
Expand this cell to see the calculation log.

The results can be pulled in from the files using this:

```
Do[detunedDipole[δ] = ReadList[filename[δ]], {δ, δRange}]
```

Or saved into a single location using this:

```
Save[FileNameJoin[{NotebookDirectory[], "Detuning scan collected data.txt"}], detunedDipole]
```

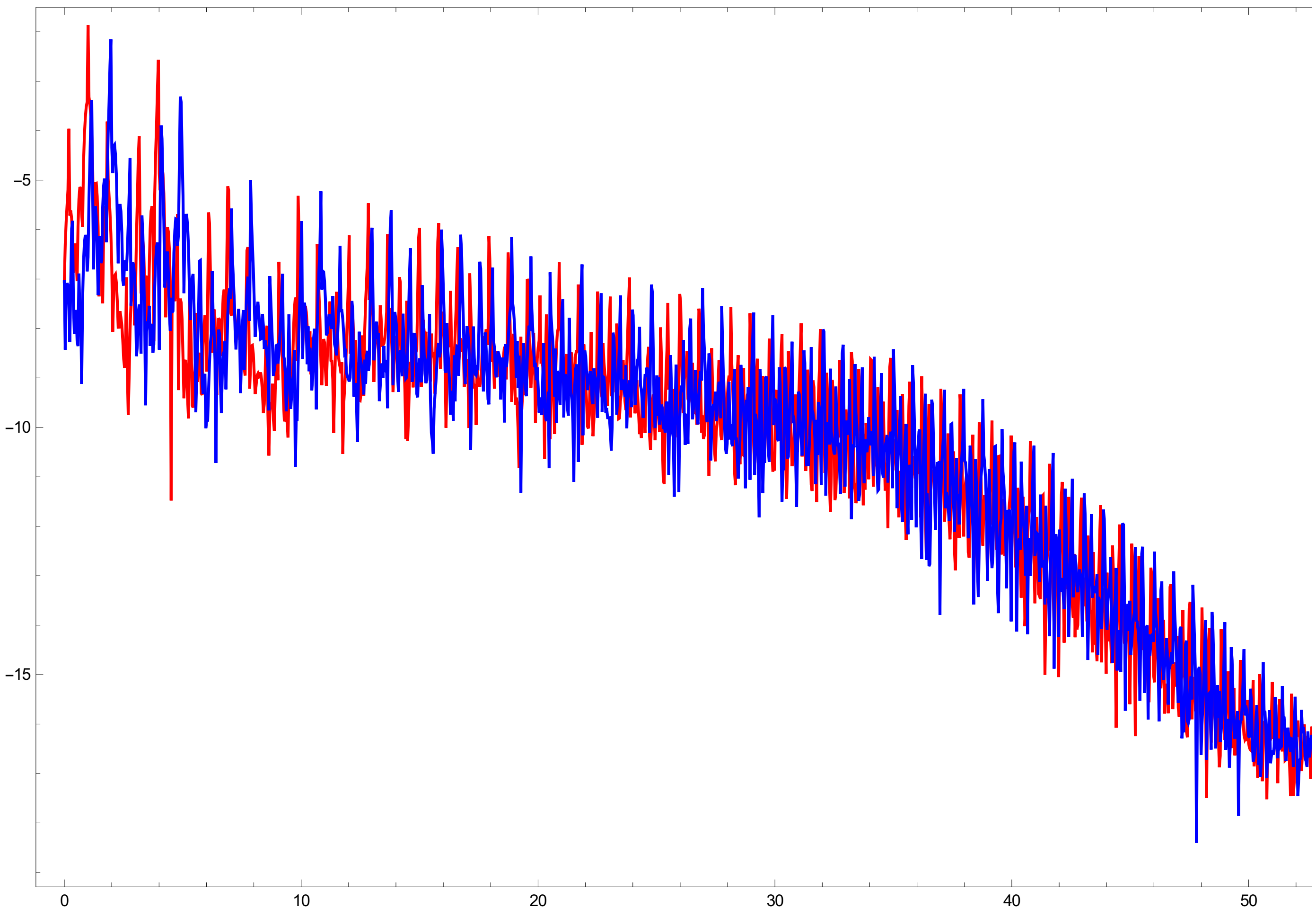
```
DumpSave[
  FileNameJoin[{NotebookDirectory[], "Detuning scan collected data.mx"}], detunedDipole];
```

and pulled in from the single location using this:

```
<< (FileNameJoin[{NotebookDirectory[], "Detuning scan collected data.txt"}]);
```

A sample spectrum looks like this:

```
biColorSpectrum[detunedDipole[0.001 RandomInteger[{0, 1000}]],
 CarrierFrequency → 45.6 / 800, TotalCycles → 3, PointsPerCycle → 115]
```



Plots from the original paper

The plots from the original paper were produced using the code below. For simplicity we pre-define an interpolation function.

```
conditions := Sequence[CarrierFrequency → 45.6 / 800, TotalCycles → 26, PointsPerCycle → 115]
```

```
Remove[detuningInterpolation]
With[{length = Length[getSpectrum[detunedDipole[0.], Polarization → {1, ⅈ}]]},
 AbsoluteTiming[
  Table[
    detuningInterpolation[ϵ] = Interpolation[
      Flatten[Table[
        {{
            harmonicOrderAxis[TargetLength → length, conditions],
            Table[δ, {length}]
           }^T,
          Log[10, getSpectrum[detunedDipole[δ], Polarization → {1, ϵ ⅈ}]]
         }^T
        , {δ, δRange}], 1]]
    , {ϵ, {1, -1}}];
 ]
]
{2.99829, Null}
```

Some plotting admin:

```
CMRmap = Function[x, Blend[{⬛, 🟦, 🟪, 🟪, 🟥, 🟧, 🟨, 🟨, ⬜}, x]];
CMRwithMin[minIn_, minOut_: 1. / 9] :=
 Function[x, CMRmap[If[x < minIn, minOut/minIn x, minOut + (1 - minOut) (x - minIn)/(1 - minIn)]]]
min = 6. × 10^-9;
max = 5. × 10^-7;
colorfunction = CMRwithMin[min / max];
HOTicks[ϵ_] :=
 ({#, If[ϵ == 1, Style[#, Black], ""], {0.02, 0}, {Thickness[0.005], Gray}} & /@ Range[12, 18, 1]) ~
  Join~ ({#, "", {0.01, 0}, {Thickness[0.004], Gray}} & /@ Range[11 + 1/2, 18 + 1/4, 1 / 4])
downTicks = {{0, Style[0, Black], 0}, {0.25, Style[0.25, Black], 0}}~Join~
   ({#, Style[#, Black], {0.015, 0}, {Thickness[0.005], Gray}} & /@ Range[0.05, 0.20, 0.05]) ~
   Join~ ({#, "", {0.01, 0}, {Thickness[0.004], Gray}} & /@ Range[0.01, 0.24, 0.01]);
upTicks = ({#, "", {0.015, 0}, {Thickness[0.005], Gray}} & /@ Range[0.05, 0.20, 0.05]) ~
   Join~ ({#, "", {0.01, 0}, {Thickness[0.004], Gray}} & /@ Range[0.01, 0.24, 0.01]);
```

The plot itself:

```
Row[Table[
  splittingsScan[ϵ] = RegionPlot[
    True
    , {δ, 0, 0.25}, {HO, 11.25, 18.5}
    , AspectRatio → 1.2
    , PlotRangePadding → None
    , ImagePadding → 1 {{35 + 15 ϵ, 20}, {70, 6}}
    , ImageSize → {Automatic, 550}
    , PlotPoints → 600
    , FrameStyle → Automatic
    ,
    FrameLabel → {Style["ω'/ω-1", Black, 12], If[ϵ == 1, Style["Harmonic Order", Black, 16], ""]}
    , ColorFunctionScaling → False
    , FrameTicks → {{HOTicks[1], HOTicks[-1]}, {downTicks, upTicks}}
    , ColorFunction → Function[{δ, HO}, colorfunction[10^detuningInterpolation[ϵ][HO, δ]/max]]
    , PlotLabel →
     Style[StringJoin[ϵ /. {1 → "Right", -1 → "Left"}, "-circular harmonics"], Black, 16]
   ]
  , {ϵ, {1, -1}}]
]
```

(Removed to keep file size low.)