CLASS-OneLoop
Numerical aspects and outlook

Julien Lesgourgues (TTK, RWTH Aachen university)

Theoretical Modeling of Large-Scale Structure of the Universe, Edinburgh, 4.06.2024

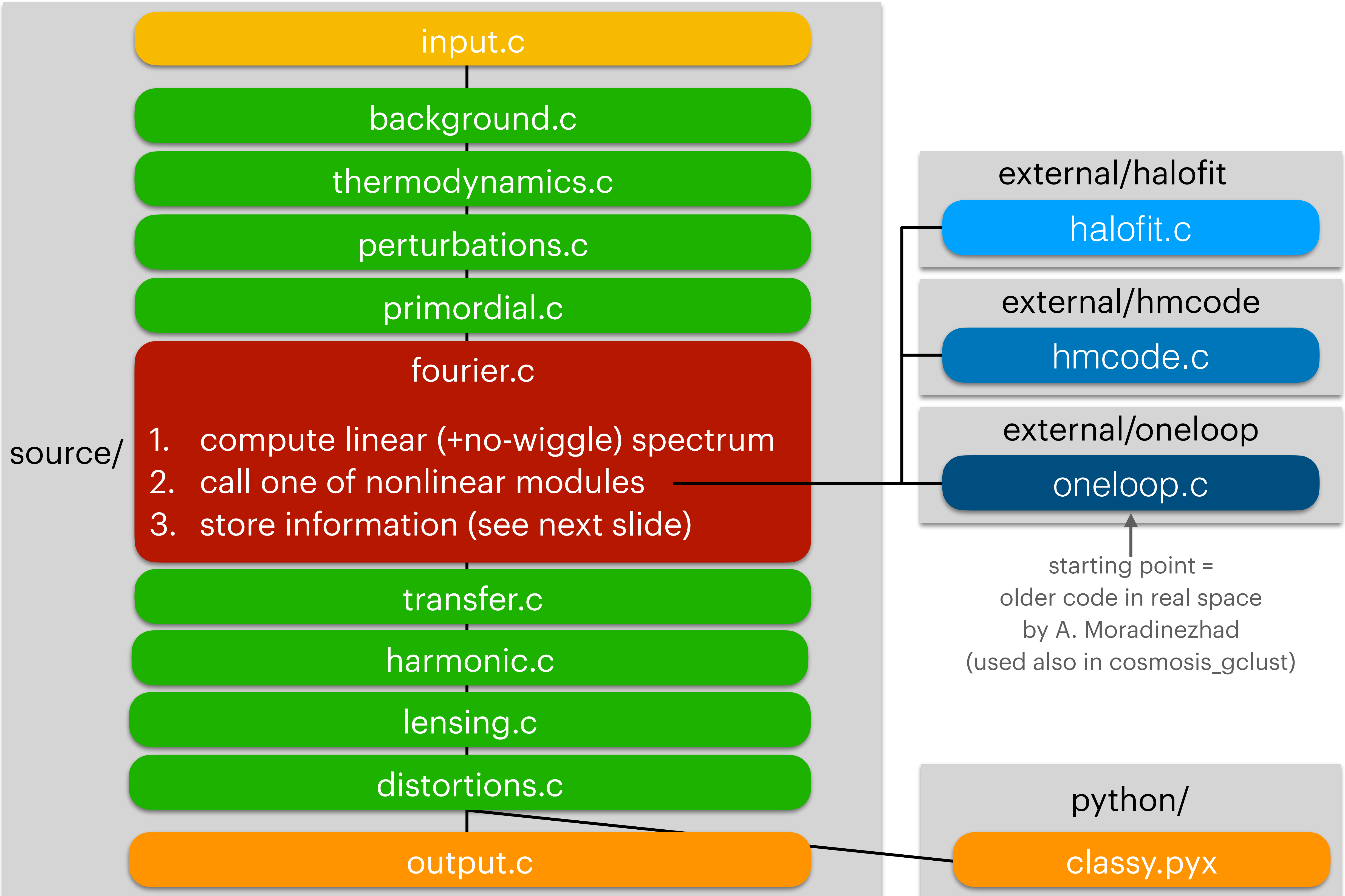
collab. with Dennis Linde, Azadeh Moradinezhad, Christian Radermacher, Santiago Casas,

arXiv 2402.09778, accepted in JCAP

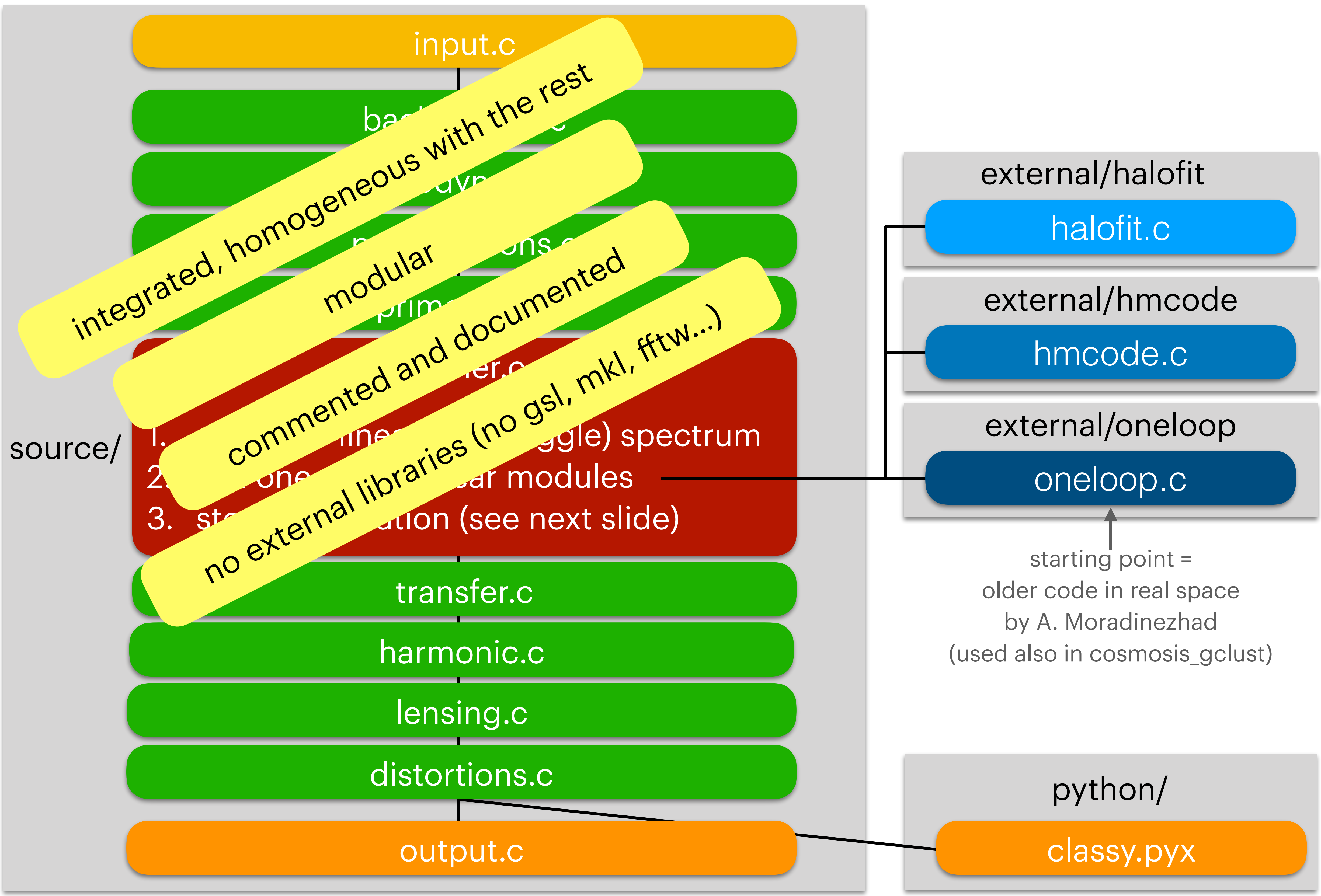
In preparation: release of the code in CLASS v3.4 + release paper with tests and documentation

Final version still under development, input/suggestions/requests welcome!

Overall structure of CLASS v3.4



Overall structure of CLASS v3.4



What happens in the oneloop module ?

- log-Fourier approach with oneloop_integration = log_fourier (also: direct_integration)
  - if (cosmology-independent) kernels K_{ij}^n not found in binary files or cached in memory, compute them and write them, otherwise read them
  - log-Fourier transform linear spectrum $P_{\rm lin}$ into coefficients c_i at selected z_k
  - compute 42 loops $L^{(n)} = c_i \, K_{ij}^{(n)} \, c_j$ at these z_k

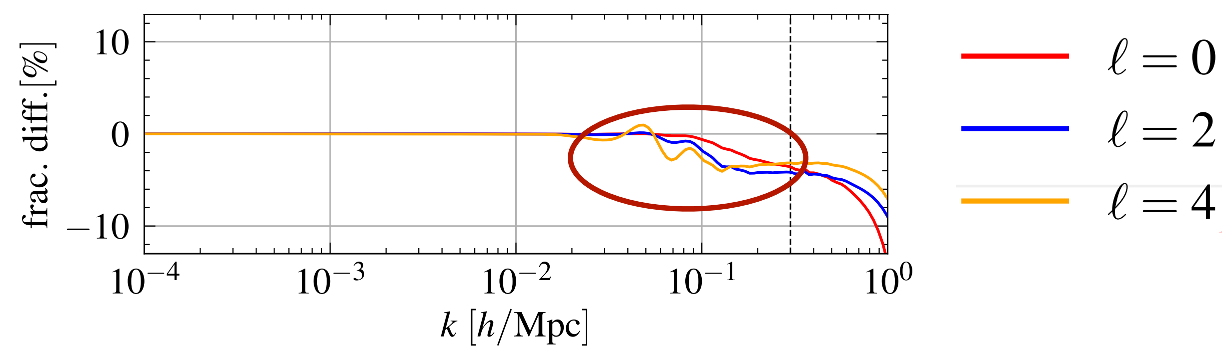
can be just zero $z = 0$,
and then scaling with $D(z,k)$, $f(z,k)$,
or several values

  - oneloop_strategy = store_spectra [stick to old CLASS logic]
    - uses input {biases} {counter-terms} {stoch. terms} to store $P_X^{\rm real}(k,z)$, $P_X^{\rm rsd}(k,z,\mu)$, $P^{\rm rsd}_{X,\ell=0,2,4}(k,z)$ for $X \in$ {matter, tracer (e.g. galaxies), cross} at tabulated values
    - observables can be retrieved at any z within output.c or classy.pyx using interpolation

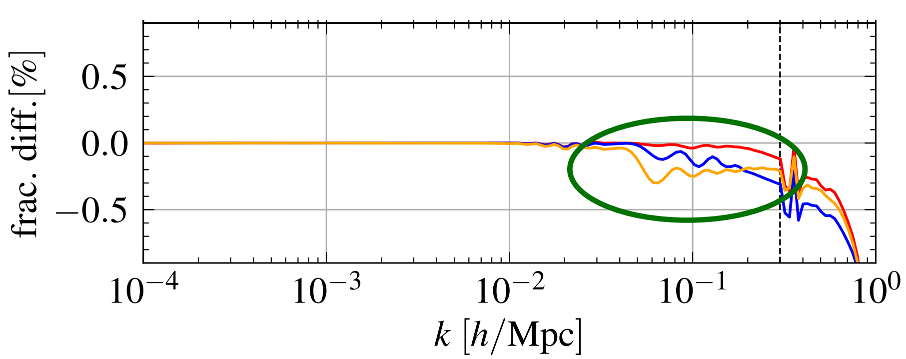
  - oneloop_strategy = store_loops [fast/slow parameters in MCMC]
    - only store μ-independent $\{L^{(n)}\}$ at each z_k
    - in classy.pyx, fast functions can build observables on demand for requested {bias} {counter-terms} {stoch. terms} $\{z\}$ $\{\mu\}$

What is the numerical accuracy of the oneloop module?

- Comparisons in $k \in [0.01, 0.3]\, h/\mathrm{Mpc}$ for $N_{\mathrm{FFT}} = 256$
- Direct integration vs. FFTlog (see 2402.09778 Fig 1):
  - Difference in each individual loop contribution < 0.1%
- CLASS-OneLoop vs. CLASS-PT [Chudaykin, Ivanov, Philcox, Simonovic 20] (see 2402.09778 Fig. 2,3,4):
  - Different de-wiggling algorithms (Gaussian filter vs. spectral decomposition): < 0.5% in real space, < 5% for multipoles; could be absorbed by counter-terms...



  - Same de-wiggling algorithms (Gaussian filter): < 0.2% in real space, < 0.3% for multipoles



$b_1 = 1.8, \quad b_2 = -0.5,$

$b_{\mathcal{G}_2} = -0.05, \quad b_{\Gamma_3} = 0.08$

- Also consistent with cosmosis_gclust [Moradinezhad et al.], velocilaptors [Chen et al. 20], PyBird [D'Amico et al. 20]

How fast is the oneloop module?

- time flow [PRELIMINARY] on MacBookPro Intel i9 2.3GHz 16 cores:
  - $\Lambda\mathrm{CDM} + m_\nu$, no CMB, single $\{z_k\} = 0$
  - $P_{\rm lin}$ computed till $k_{\rm max} = 50\,h/\mathrm{Mpc}$ and extrapolated till $k_{m...} = 10^3\,h/\mathrm{Mpc}$ ($\times\,4$ for tracers)
  - Request: spectrum $P^{\rm rsd}_{\rm tracer}(k, \mu, z)$ for array of 3 z, 137 k,

slower version

Times in [s] , N_c = # of cores	FFTlog $N_{FFT} = 256$
kernels ***K*** (once per MCMC)	$73/N_c = 4.5$
log-Fourier transform P_{lin} into ***c*** (once per cosmology and z_k)	$0.005/N_c \sim 10^{-4}$
individual loops (***L*** = ***c K c***) (once per cosmology and z_k)	$2.1/N_c = 0.13$
build spectrum from loops (scale with # of output z, here 3)	$0.006/N_c \sim 10^{-4}$
rest of CLASS with $N_c = 16$	0.5
total CLASS with $N_c = 16$ (cached kernels)	0.6

→ log-Fourier transform: scales like $N_{\rm FFT}\ln(N_{\rm FFT})$

→ individual loops: scales like $N^2_{\rm FFT}$

Can we do better?

- Yes, at least by revisiting the log-Fourier transform...
- logFT of $P_{\rm lin}$: $N_{\rm FFT}$ coefficients $c_j = \frac{1}{T}\int_{\ln(k_{\rm min})}^{\ln(k_{\rm max})} d\ln k \; P_{\rm lin}(k) \; \exp\left[\left(\frac{2\pi i j}{T} - \nu\right)\ln(k)\right]$
- fourier_mode = fourier_mode_fft (FFTlog)
  - discrete Fast Fourier Transform with $N_{\rm FFT}$ values k_i , divide-and-conquer algorithm
  - (implemented in C from scratch by N. Schöneberg for arXiv:1807.09540 in tools/fft.c)
  - decrease number of coefficients $N_{\rm FFT} \Rightarrow$ decrease their accuracy
- fourier_mode = fourier_mode_spline (SFTlog)
  - spline $P_{\rm lin}(k_i) \rightarrow$ piece-wise cubic polynomial, moments $P_i'' \rightarrow c_j = \Sigma_i \, \alpha_{i,j} \, P_i''$
  - (implemented in C from scratch by C. Radermacher for this work in tools/array.c)
  - works with $P_{\rm lin}(k_i)$ sampled at non-evenly-spaced $\ln k_i$ (e.g. taken from previous modules and dense for BAO, sparse elsewhere)
  - decrease number of coefficients $N_{\rm FFT}$ with constant accuracy

Can we do better?

- time flow [PRELIMINARY] on MacBookPro Intel i9 2.3GHz 16 cores:
  - $\Lambda\mathrm{CDM} + m_\nu$, no CMB, single $\{z_k\} = 0$
  - P_lin computed till $k_\mathrm{max} = 50\,h/\mathrm{Mpc}$ and extrapolated till $k_\mathrm{max} = 10^3\,h/\mathrm{Mpc}$ ($\times\,4$ for tracers)
  - Request: spectrum $P^\mathrm{rsd}_\mathrm{tracer}(k, \mu, z)$ for array of 3 z, 137 k, 5 μ

accuracy stable at 10^{-4} level

Times in [s] , N_c = # of cores	FFTlog $N_{FFT} = 256$	SFTlog $N_{FFT} = 96$ $N_k = 301$
kernels ***K*** (once per MCMC)	$73/N_c = 4.5$	$10/N_c = 0.6$
log-Fourier transform P_{lin} into ***c*** (once per cosmology and z_k)	$0.005/N_c \sim 10^{-4}$	$0.010/N_c \sim 5\ 10^{-4}$
individual loops (***L*** = ***c K c***) (once per cosmology and z_k)	$2.1/N_c = 0.13$	$0.3/N_c = 0.02$
build spectrum from loops (scale with # of output z, here 3)	$0.006/N_c \sim 10^{-4}$	$0.006/N_c \sim 10^{-4}$
rest of CLASS with $N_c = 16$	0.5	0.5
total CLASS with $N_c = 16$ (cached kernels)	0.6	0.5

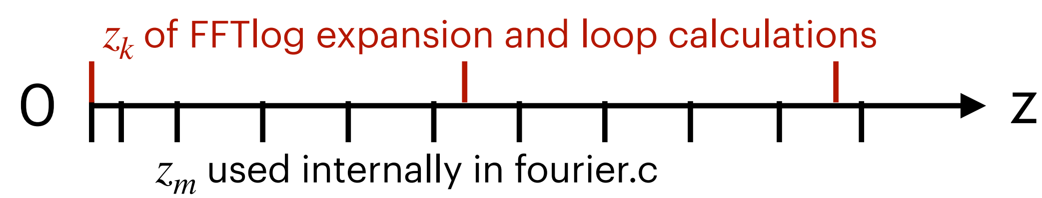
Conclusions

- Class-OneLoop already fast enough (~ 20 ms) unless $P_{\rm lin}$ calculation substituted by emulator…
- Before release of v3.4: need time to polish style (user-friendliness), set robust default precision parameters, provide clear documentation in release paper
- Possible developments -> panel discussion this evening

Back-up slides

Which redshifts are used in oneloop.c ?

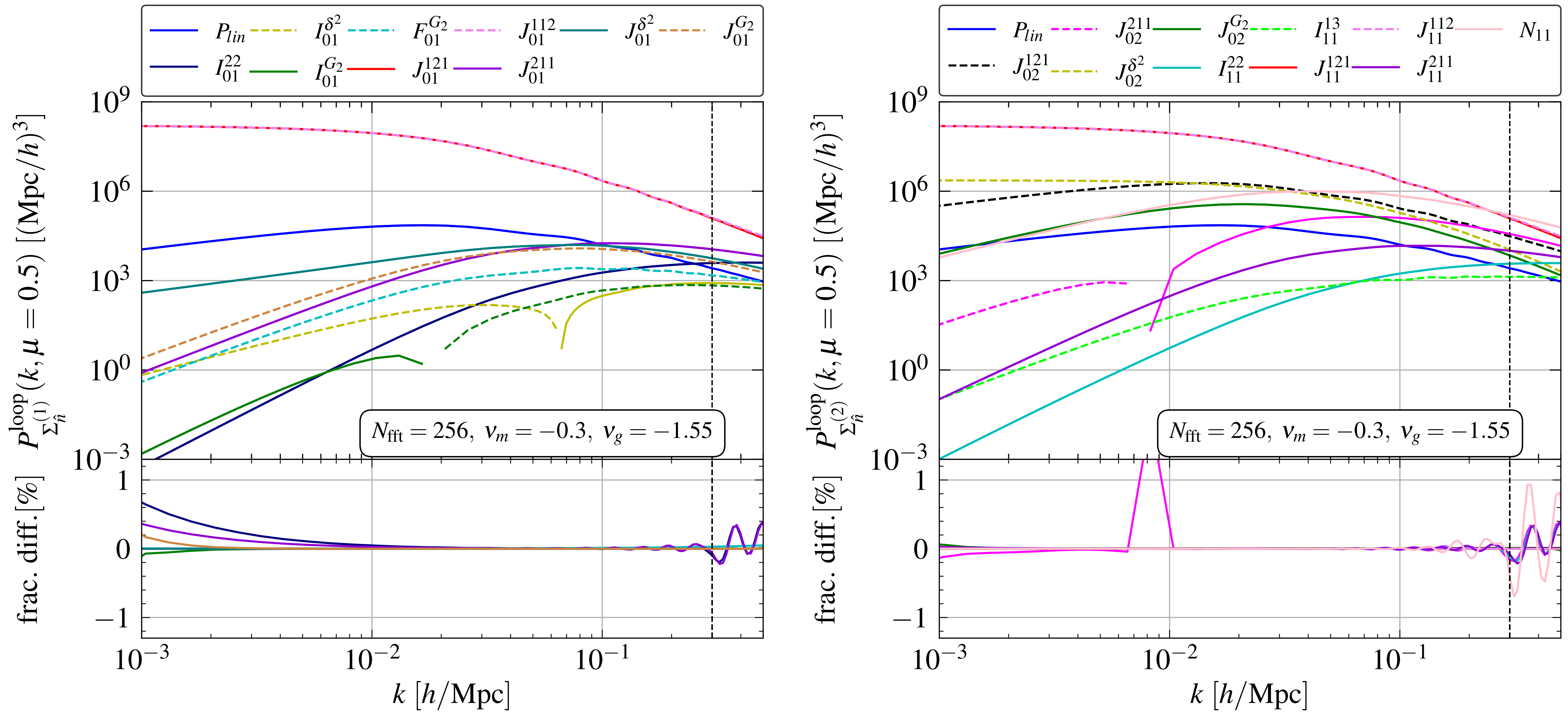
- oneloop_integration = log_fourier (also: direct_integration, uses CUBA library)
  - if (cosmology-independent) kernels K_{ij}^n not found in binary files, compute them and write them, otherwise read them
  - log-Fourier transform linear spectrum coefficients c_i at selected z_k
  - compute $n = 1, \ldots, 40$ loops $L^n = c_i K_{ij}^n c_j$ at each z_k



  - oneloop_redshift = single
    - only use $z_k = 0$
    - rescale to any other z using growth factor/rate of the model
  - oneloop_redshift = all
    - expansion/loops at each z_m used internally by fourier.c (about a hundred)
    - no rescaling, but still, kernels are z-independent...
  - oneloop_redshift = few
    - expansion/loops at z_k passed by user with "z_pk = .., , .."
    - rescaling to closest z_k using growth factor/rate of the model

Numerical accuracy of FFTlog/integration scheme?

- Direct integration versus FFTlog for loops with first / second velocity momenta

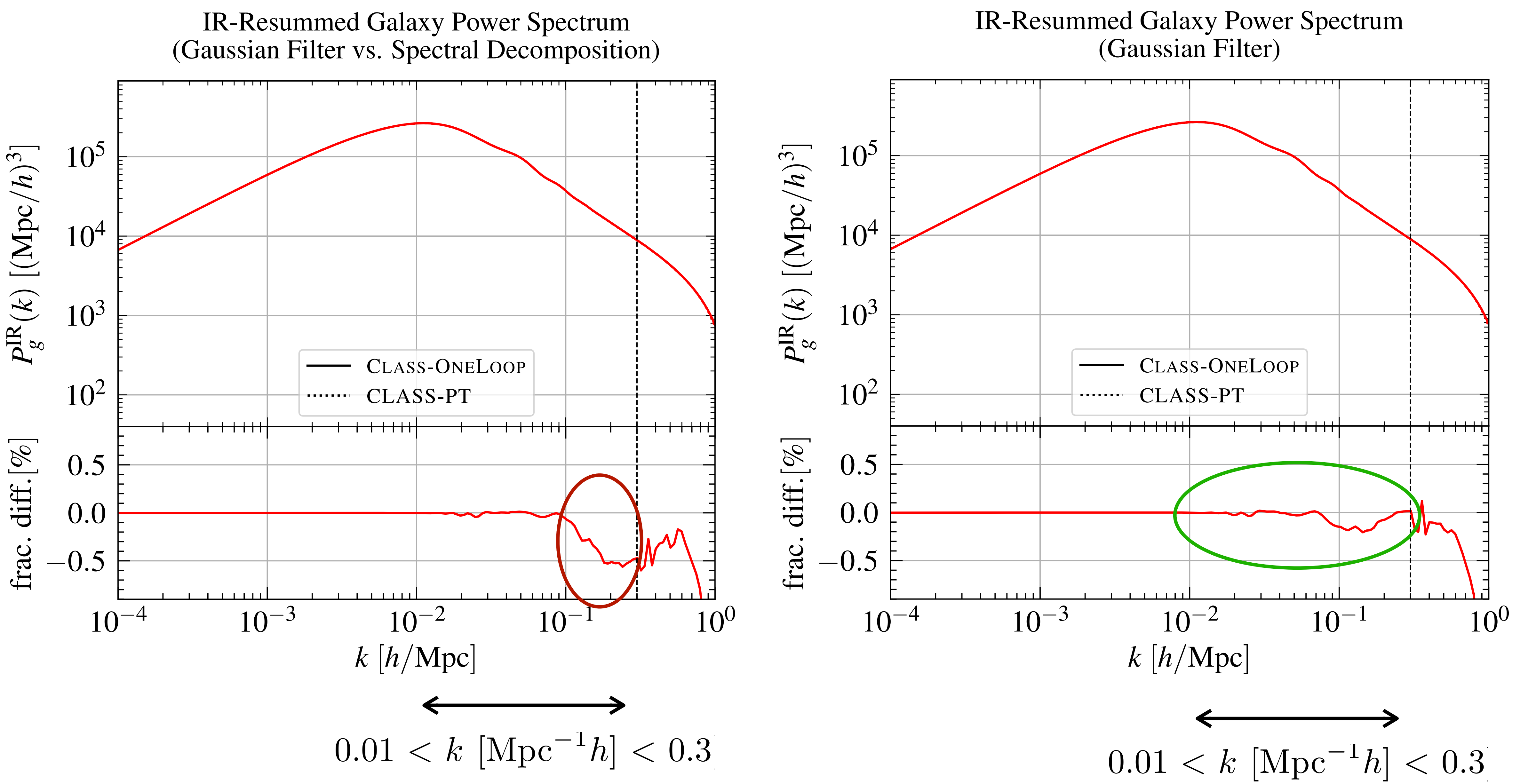


$0.01 < k\ [{\rm Mpc}^{-1}h] < 0.3$ $\qquad$ $0.01 < k\ [{\rm Mpc}^{-1}h] < 0.3$

- Always < 0.1% difference in targeted k range for $N_{\rm FFT} = 256$

Comparison with CLASS-PT?

- Comparison in real space at $z = 0$, $N_{\rm FFT} = 256$, $b_1 = 1.8$, $b_2 = -0.5$, $b_{\mathcal{G}_2} = -0.05$, $b_{\Gamma_3} = 0.08$
- Difference dominated by dewiggling method for IR resummation; absorbed by counter-terms?

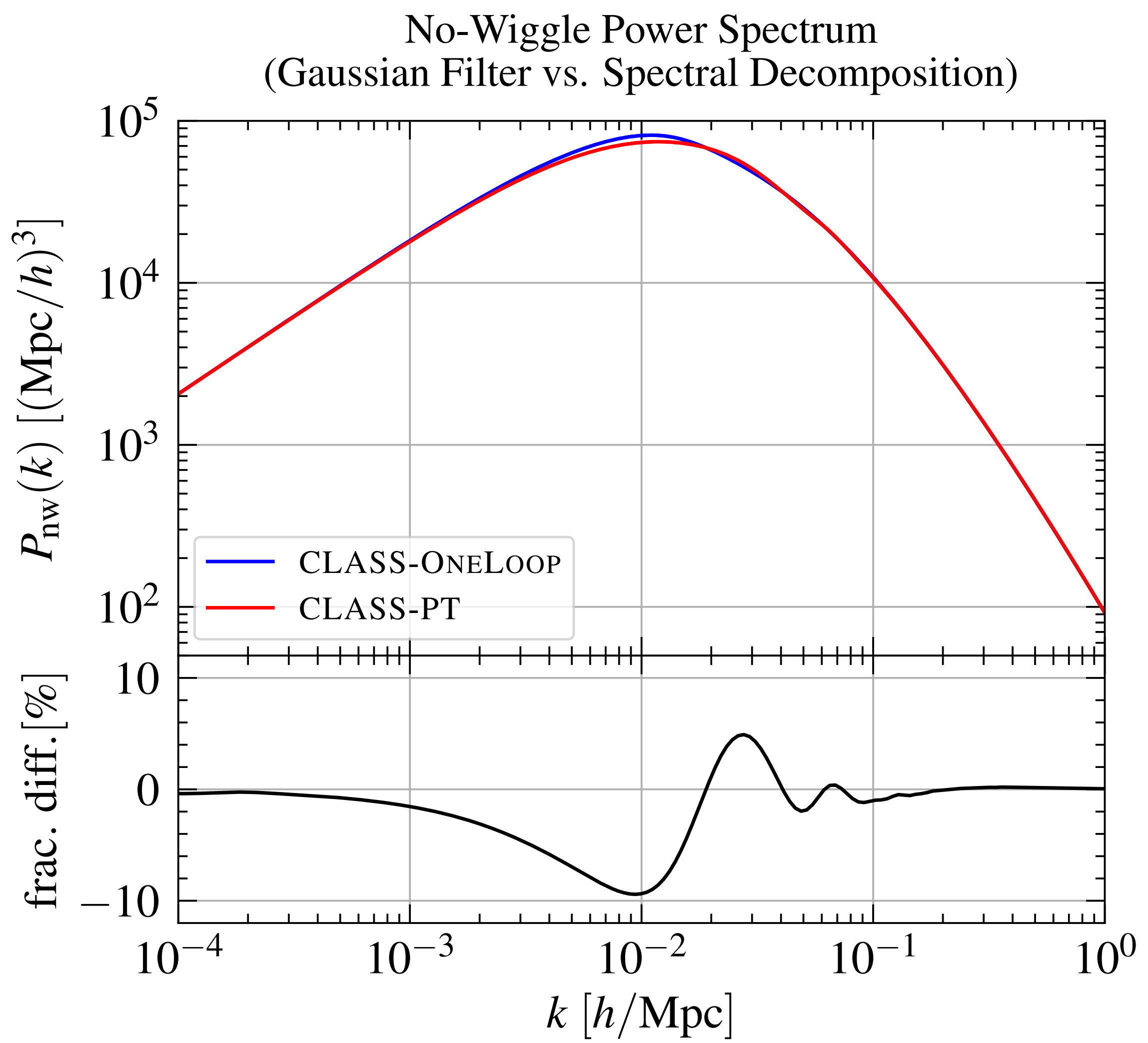


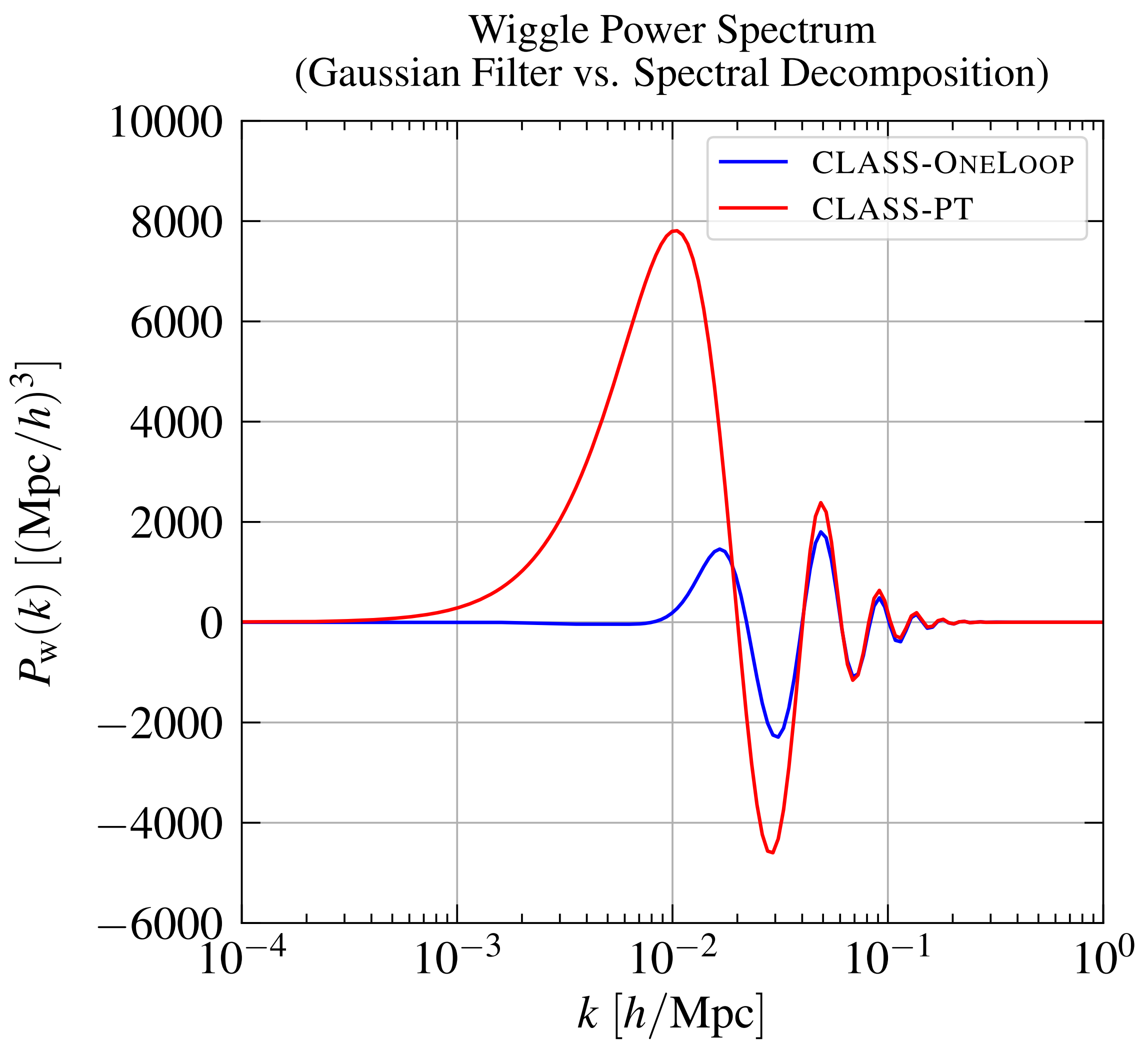
- < 0.2% when sticking to same Gaussian filter

Institute for Theoretical Particle Physics and Cosmology
RWTH AACHEN UNIVERSITY

Comparison with CLASS-PT?

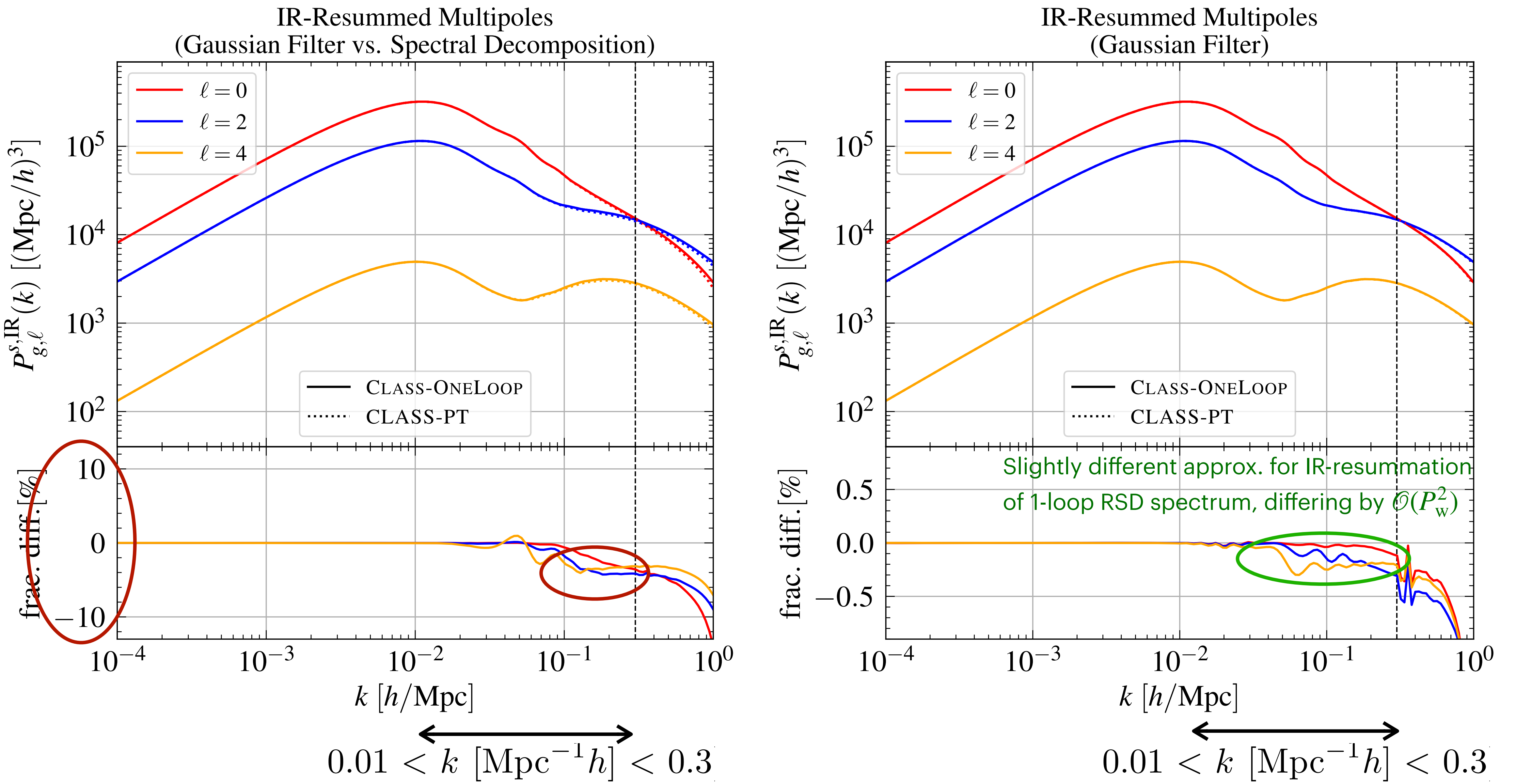
- Dewiggling method for IR resummation:
- Spectral decomposition (DST) [Hamann et al. 2010] vs. Gaussian filtering (of $P_{\rm lin}/P_{\rm HE}$)





Comparison with CLASS-PT?

- Comp. in redshift space at $z = 0$, $N_{\rm FFT} = 256$, $b_1 = 1.8$, $b_2 = -0.5$, $b_{\mathcal{G}_2} = -0.05$, $b_{\Gamma_3} = 0.08$
- Difference dominated by dewiggling method for IR resummation; absorbed by counter-terms?



- < 0.3% when sticking to same Gaussian filter

TTK Institute for Theoretical Particle Physics and Cosmology | RWTH AACHEN UNIVERSITY

Performance?

- FFTlog expansion domain: $k_{\rm min} = 10^{-6}\,h/{\rm Mpc}$, $k_{\rm max} = 10^{3}\,h/{\rm Mpc}$ ($\times$ 4 for tracers) (avoid ringing)
- use $P_{\rm lin}(k)$ extrapolation for $k \in [50, 10^3]\,h/{\rm Mpc}$ without impact on $P_{\rm oneloop}(k < 1\,h/{\rm Mpc})$
- on RWTH Aachen cluster node: 2 Intel Xeon Platinum 8160 (24 cores each), 192GB RAM
- [preliminary from paper I]: already improved, can still gain more

- if kernels are not cached:
  - computation of kernels K^n_{ij} + spectrum coefficients c_i + loops $L^n = c_i\,K^n_{ij}\,c_j$ at one z:

in seconds:

	$N_{\rm FFT} = 128$	$N_{\rm FFT} = 256$	$N_{\rm FFT} = 512$	Direct integration
4 threads	0.61 ± 0.26	2.05 ± 0.15	6.98 ± 0.14	~ 600
8 threads	0.40 ± 0.09	1.38 ± 0.12	3.51 ± 0.23	-
16 threads	0.52 ± 0.11	1.13 ± 0.13	2.00 ± 0.22	-

- If kernels are cached:
  - computation of spectrum coefficients c_i + loops $L^n = c_i\,K^n_{ij}\,c_j$ at one z:

in seconds:

	$N_{\rm FFT} = 128$	$N_{\rm FFT} = 256$	$N_{\rm FFT} = 512$
4 threads	0.101 ± 0.008	0.400 ± 0.003	1.467 ± 0.085
8 threads	0.046 ± 0.004	0.212 ± 0.018	0.776 ± 0.037
16 threads	0.028 ± 0.003	0.105 ± 0.003	0.382 ± 0.0

  - slightly smaller than to rest of CLASS!

Which log-Fourier Transform algorithms?

- No external libraries (only optionally CUBA if direct integration required)
- logFT of P_{lin} : $c_n = \frac{1}{T} \int_{\ln(k_{\min})}^{\ln(k_{\max})} d\ln k \; I$
- fourier_mode = fourier_mode_fft
  - discrete Fast Fourier Transform
  - scales like $N_{\text{FFT}} \ln(N_{\text{FFT}})$, but tl
  - implemented in c from scratch
- fourier_mode = fourier_mode_splin
  - $P_{\text{lin}}(k_i)$ always splined for interp
  - c_n = sum of piece-wise analytic ... ɔrs
  - works with $P_{\text{lin}}(k_i)$ sampled at r ... re)
  - scales like $N_{\text{FFT}} \times N_{\text{k}}$, but then loop calculation still scales like N_{FFT}^2
  - decorrelates N_{FFT} from N_k samples. N_{FFT} can be reduced without degrading c_n precision.
  - implemented in c from scratch by C. Radermacher for this work in tools/array.c