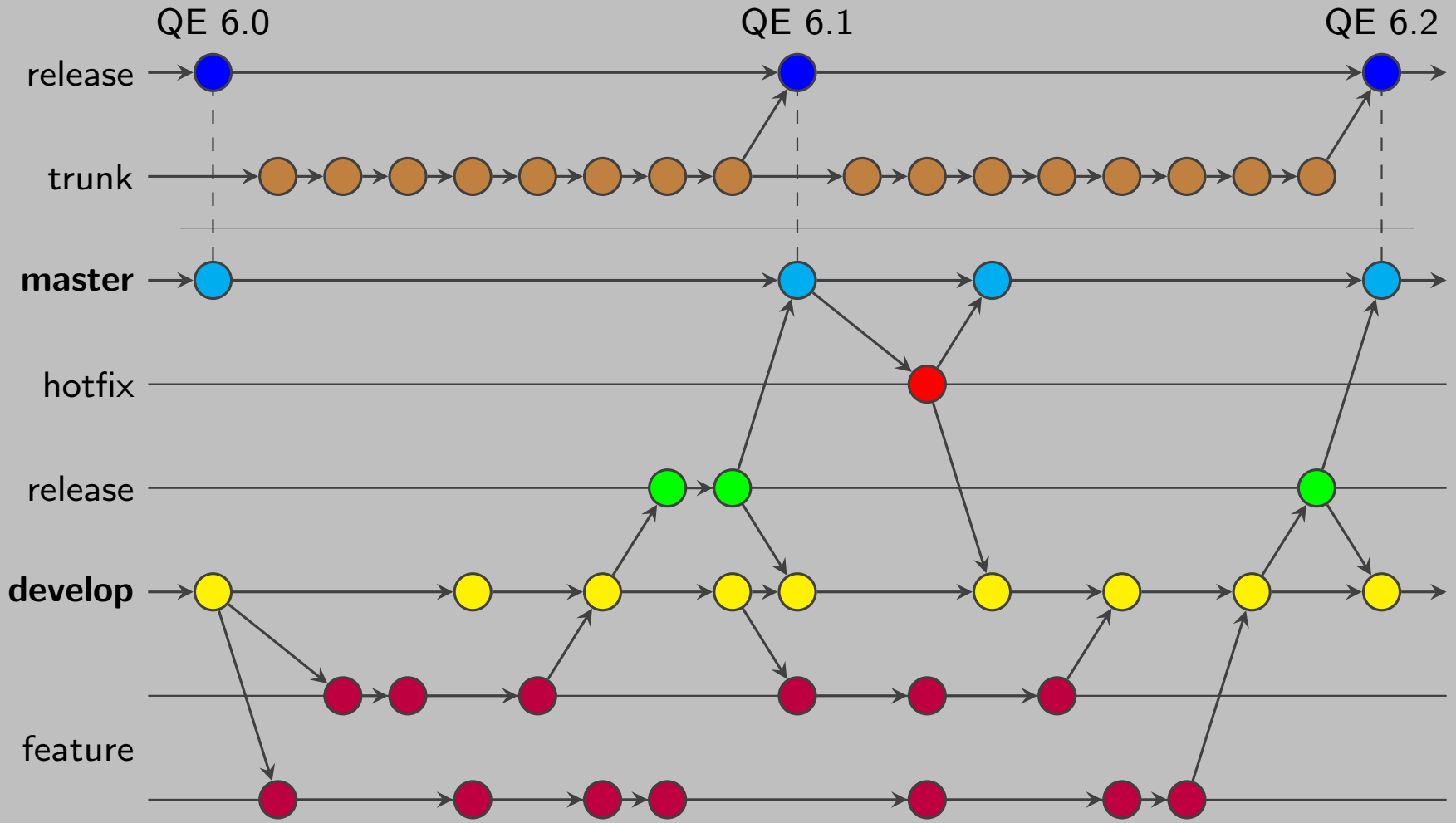
SternheimerGW Developments in 2017

Martin Schlipf

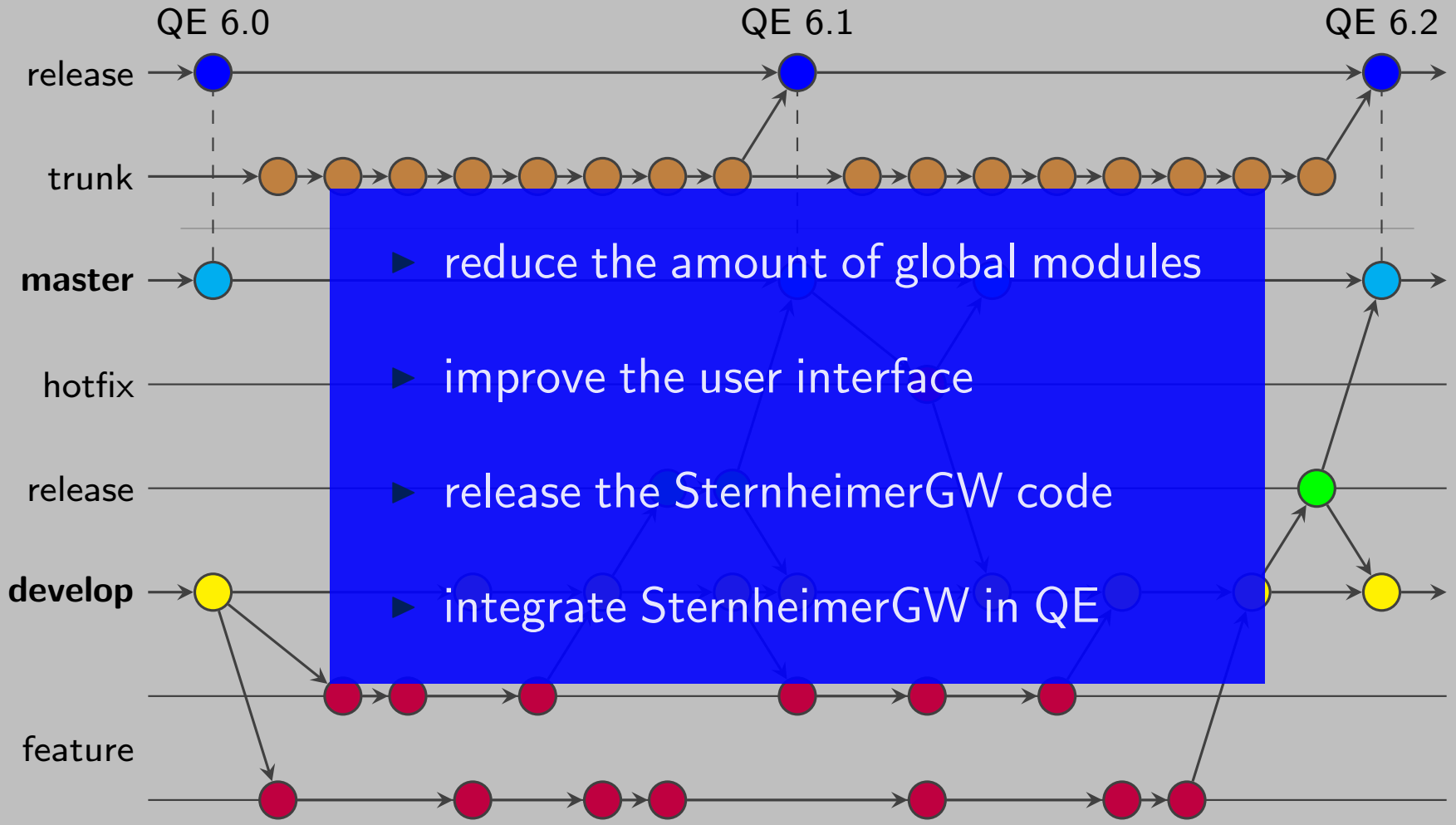
QE Developers Meeting 2018-02-01

Plan at beginning of 2017



http://nvie.com/posts/a-successful-git-branching-model/

Plan at beginning of 2017

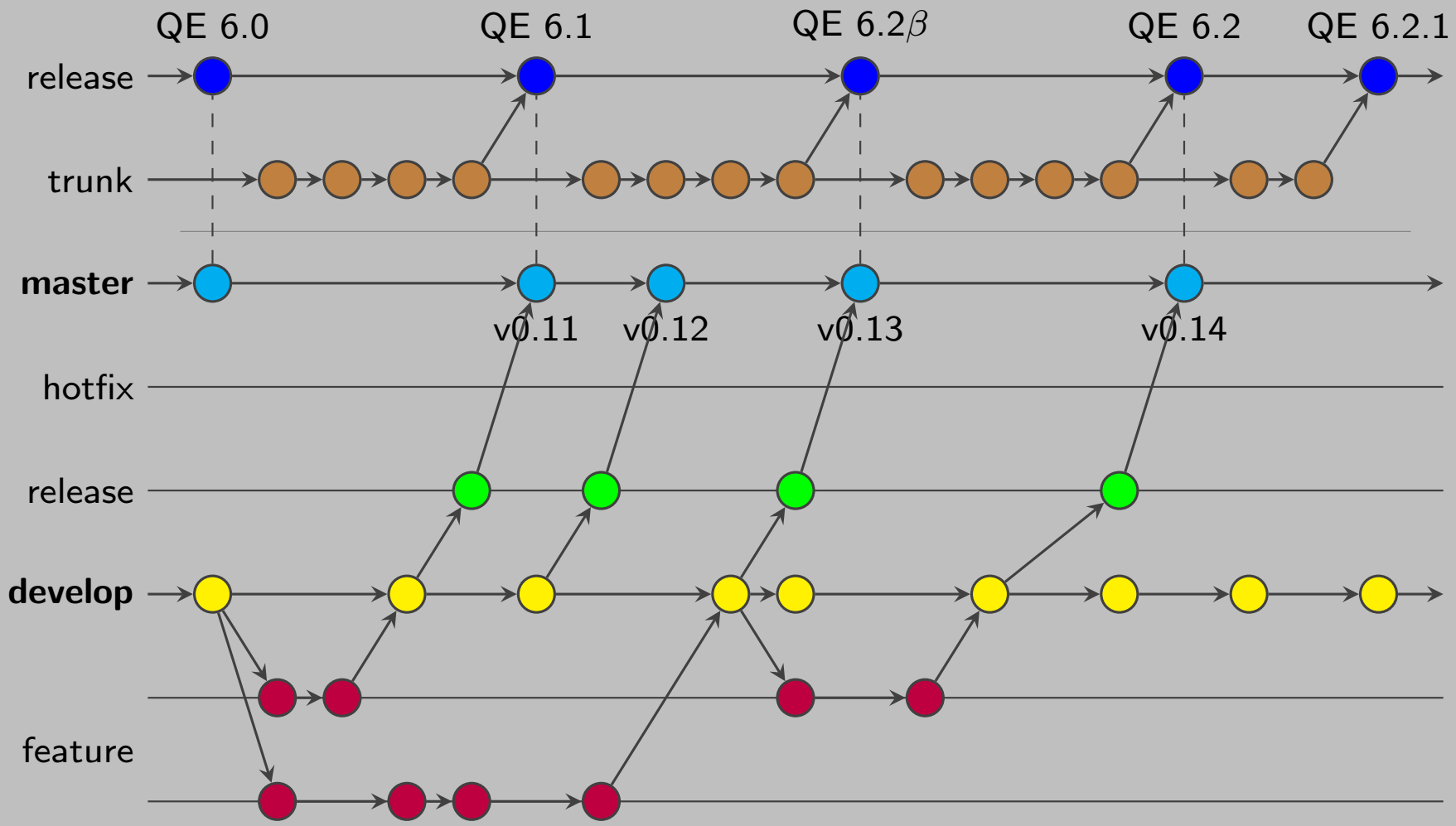


http://nvie.com/posts/a-successful-git-branching-model/

Materials Modelling & Design Group

UNIVERSITY OF OXFORD

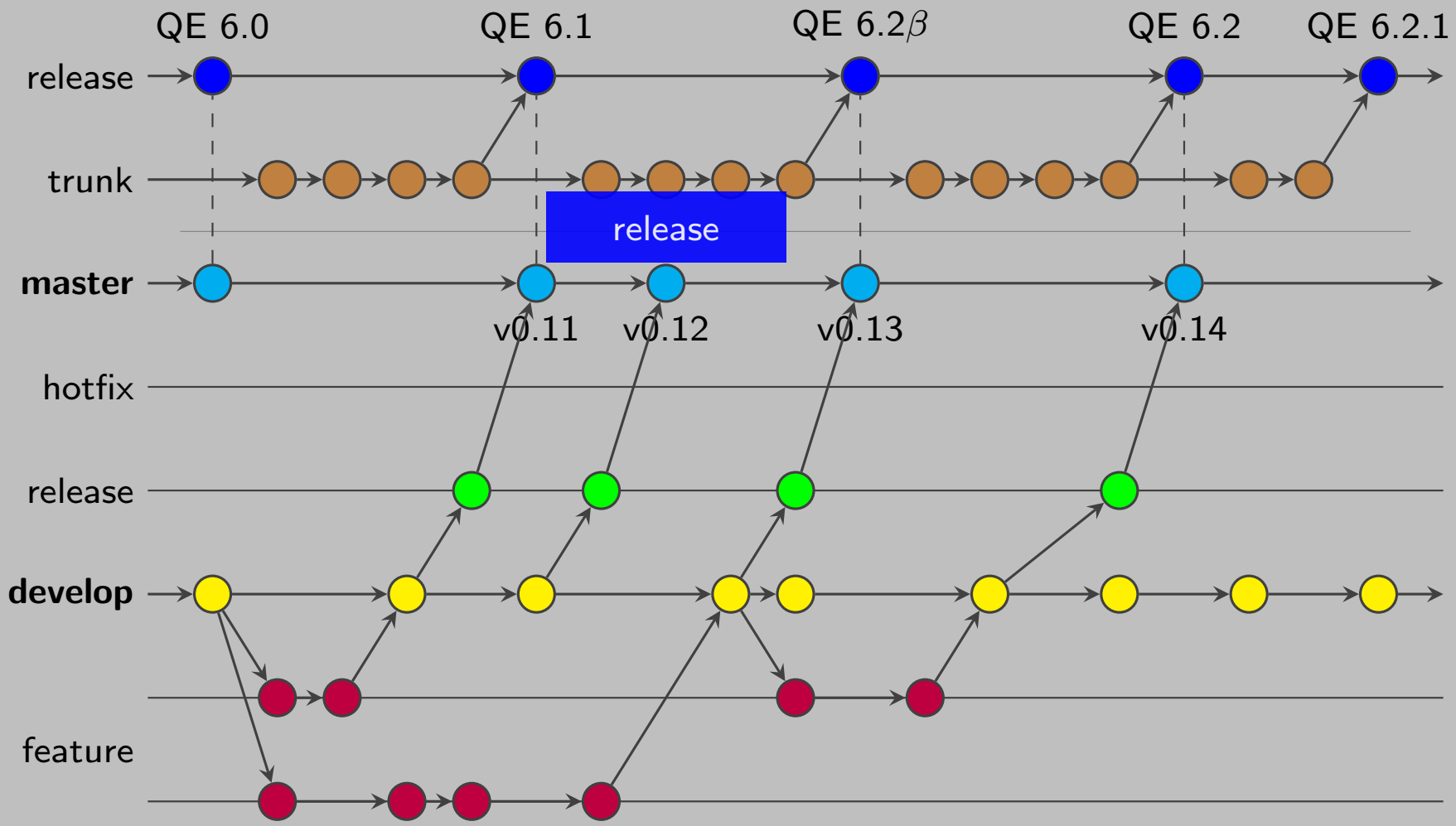
Reality in 2017



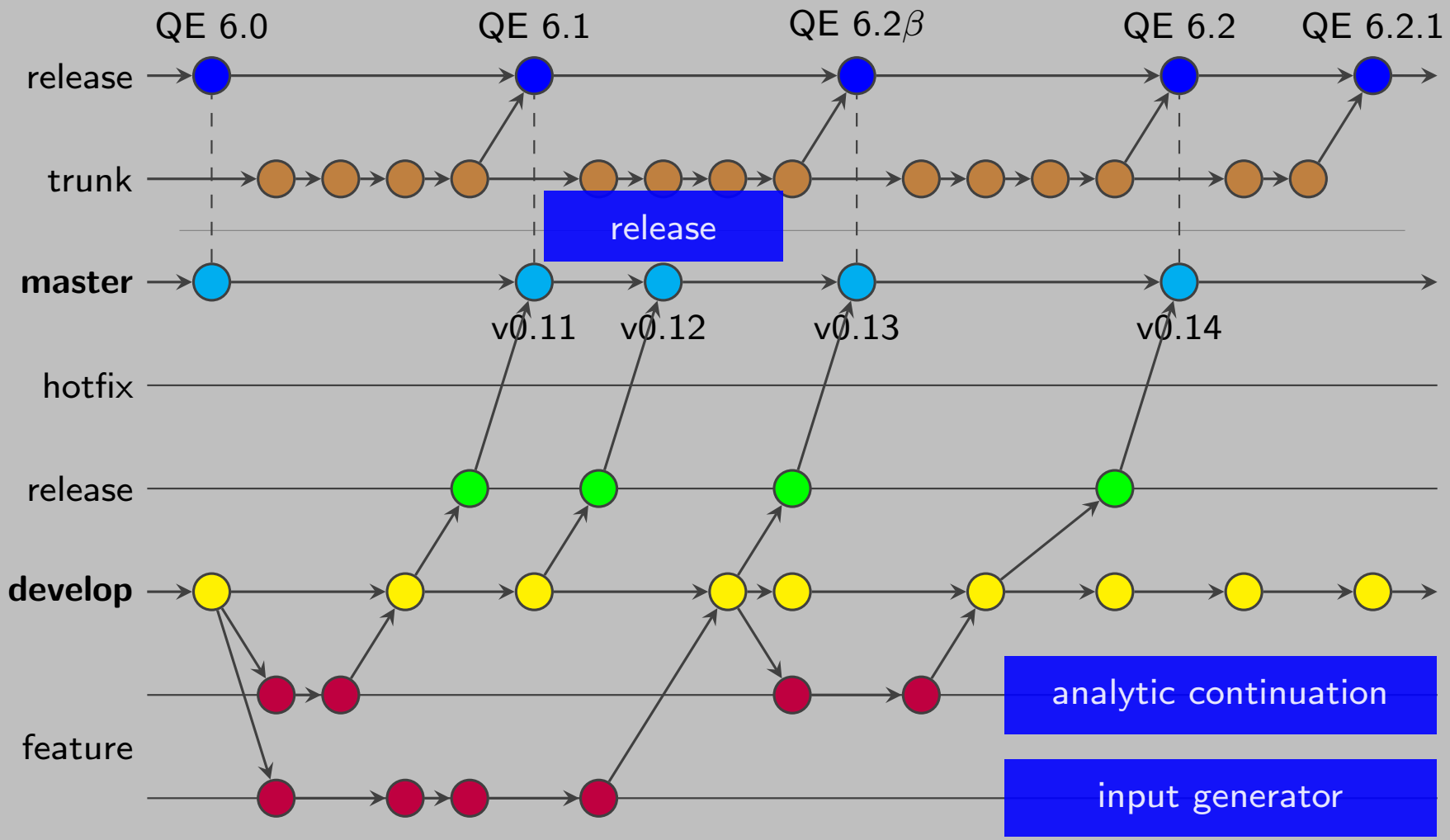
Materials Modelling & Design Group

UNIVERSITY OF OXFORD

Reality in 2017



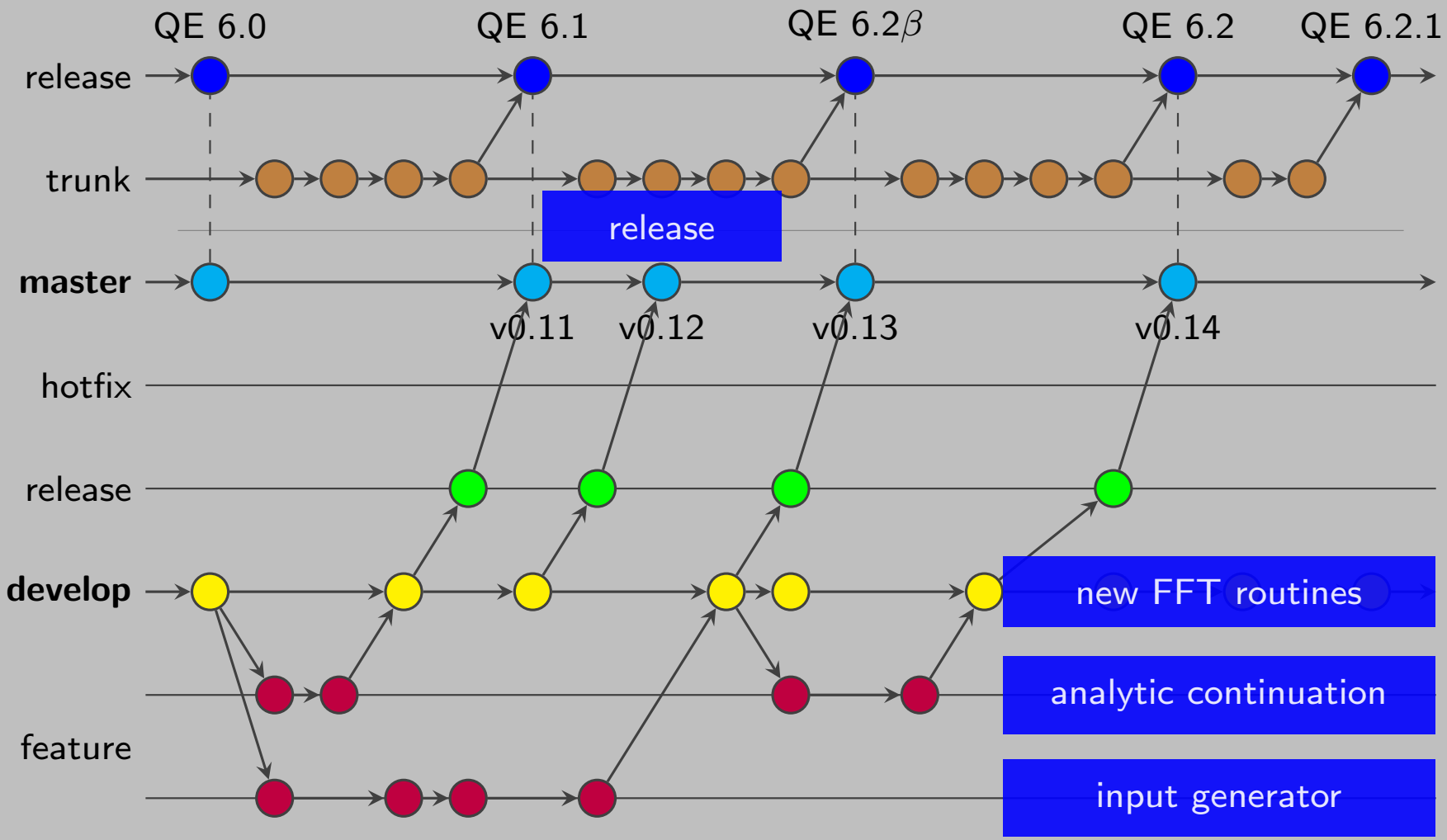
Reality in 2017



Materials Modelling & Design Group

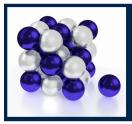
UNIVERSITY OF OXFORD

Reality in 2017



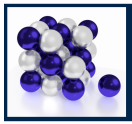
Release of the code

- Documentation: `http://sternheimergw.org/`
- Code: `https://github.com/QEF/SternheimerGW/`
- Automatic build via QE Makefile
- Improved user interface



User documentation of input variables

```
solve_coul:
  type:          character(len=256)
  default:       'direct'
  description:   Specify which method is used to solve
                 for the screened Coulomb interaction
                 'direct' determine the dielectric
                 function and invert
                 'iter' or 'iterative' to determine
                 the inverse of the dielectric
                 function iteratively which requires
                 no matrix inversion and hence less
                 memory
```



Automatic generation of input reader

part 1 – automatic types

namelist_name:

TYPE *namelist_name*_type

variable_name$_1$:
type: *type_name*$_1$
default: *default_val*$_1$
description: *comment*$_1$

variable_name$_2$:
type: *type_name*$_2$
default: *default_val*$_2$
description: *comment*$_2$

END TYPE *namelist_name*_type

Automatic generation of input reader

part 1 – automatic types

namelist_name:

variable_name$_1$:
type: *type_name*$_1$
default: *default_val*$_1$
description: *comment*$_1$

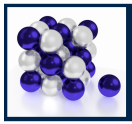
variable_name$_2$:
type: *type_name*$_2$
default: *default_val*$_2$
description: *comment*$_2$

TYPE *namelist_name*_type

!> *comment*$_1$
type_name$_1$ :: *variable_name*$_1$

!> *comment*$_2$
type_name$_2$ :: *variable_name*$_2$

END TYPE *namelist_name*_type

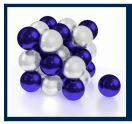


Automatic generation of input reader

part 2 – automatic interface

SUBROUTINE gw_input_read(*namelist_name*_t)

!> user input for *namelist_name*
TYPE(*namelist_name*_type), INTENT(OUT) :: *namelist_name*_t



Automatic generation of input reader

part 2 – automatic interface

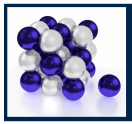
SUBROUTINE gw_input_read(*namelist_name*_t)

!> user input for *namelist_name*
TYPE(*namelist_name*_type), INTENT(OUT) :: *namelist_name*_t

!> $comment_1$
$type\_name_1$ :: $variable\_name_1$

!> $comment_2$
$type\_name_2$ :: $variable\_name_2$

NAMELIST /*namelist_name*/ $variable\_name_1$, $variable\_name_2$
⋮



Automatic generation of input reader

part 3 – automatic reading

$$\vdots$$
$$variable\_name_1 = default\_val_1$$
$$variable\_name_2 = default\_val_2$$

Automatic generation of input reader

part 3 – automatic reading

$\vdots$

$variable\_name_1 = default\_val_1$
$variable\_name_2 = default\_val_2$

READ(stdin, NML=$namelist\_name$)

$namelist\_name\_t\%variable\_name_1 = variable\_name_1$
$namelist\_name\_t\%variable\_name_2 = variable\_name_2$

END SUBROUTINE gw_input_read

Automatic generation of input reader

part 4 – automatic broadcast

SUBROUTINE gw_input_bcast(*namelist_name*_t)

!> contains the user input in namelist gw_input
TYPE(*namelist_name*), INTENT(INOUT) :: *namelist_name*_t

CALL mp_bcast(*namelist_name*_t%*variable_name*$_1$, &
meta_ionode_id, world_comm)
CALL mp_bcast(*namelist_name*_t%*variable_name*$_2$, &
meta_ionode_id, world_comm)

END SUBROUTINE gw_input_bcast

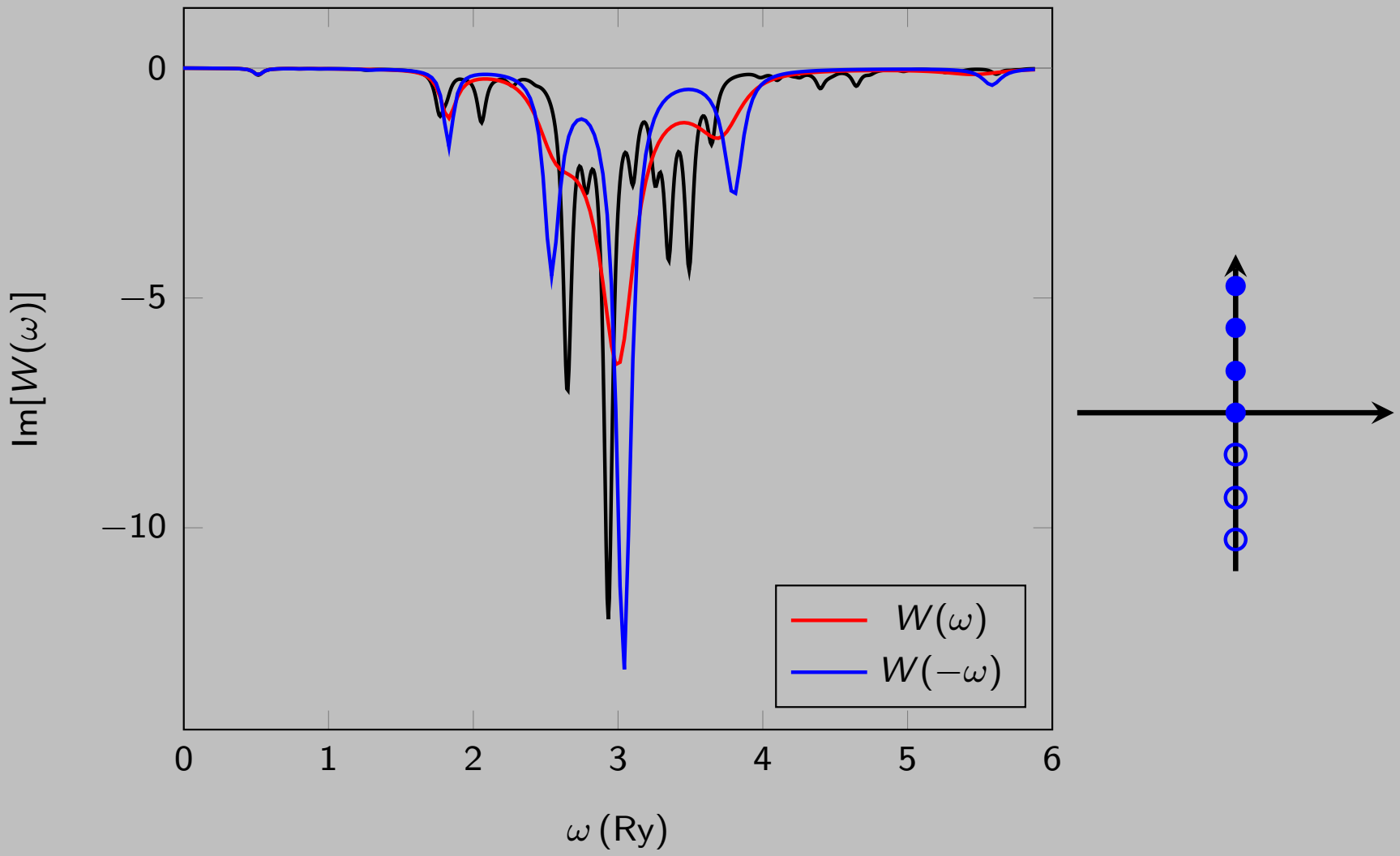
Analytic continuation

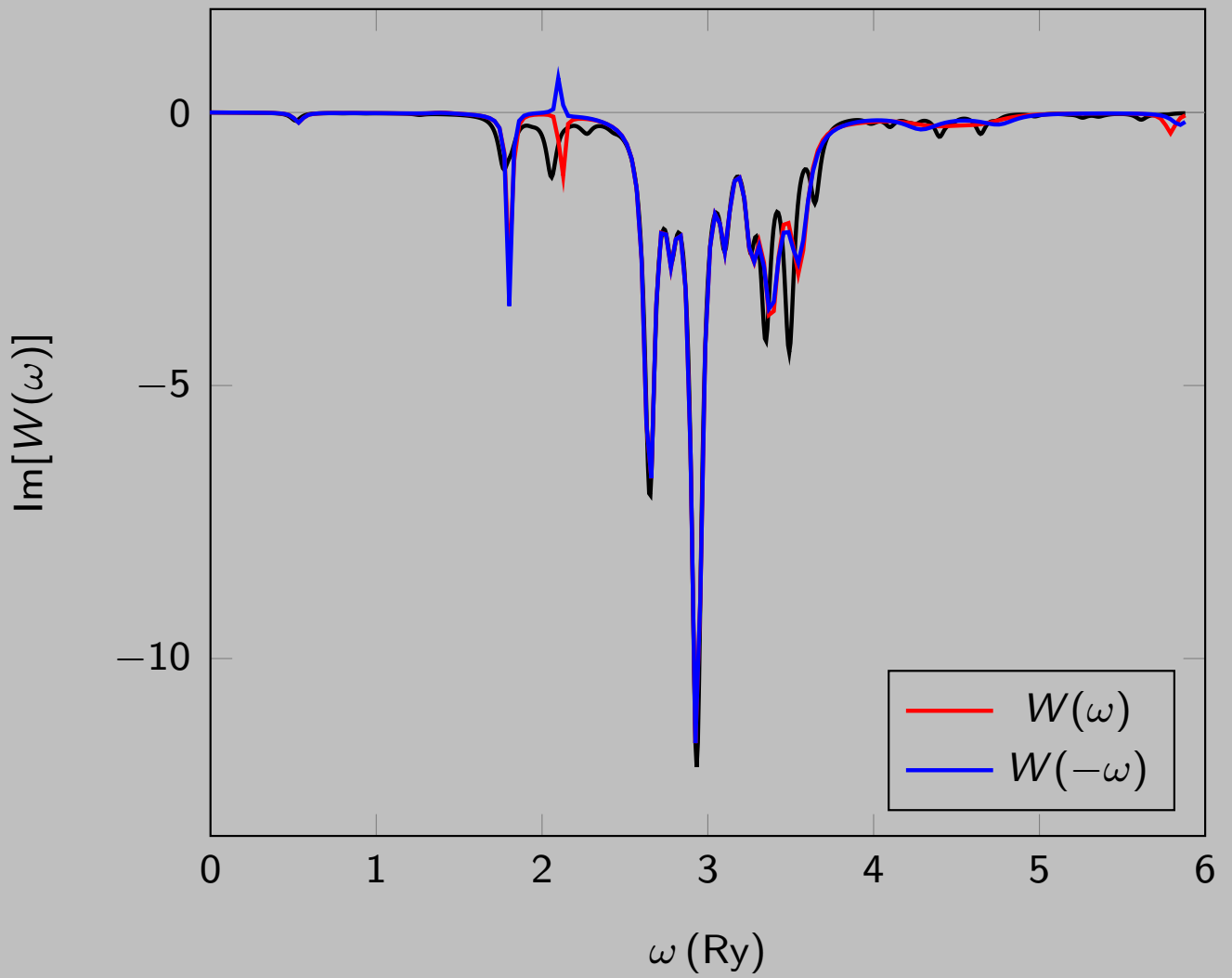
- symmetry for screened Coulomb interaction

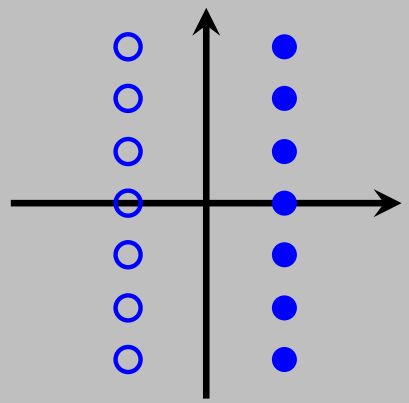
$$W(\omega) = W(-\omega)$$

- conventional Padé approximation

$$W(\omega) = \frac{\sum_m w_m \omega^m}{\sum_n w_n \omega^n}$$


0
−5
−10
Im[W(ω)]
0
1
2
3
4
5
6
ω (Ry)
W(ω)
W(−ω)


0
−5
−10
Im[W(ω)]
0
1
2
3
4
5
6
ω (Ry)
W(ω)
W(−ω)



Analytic continuation

- symmetry for screened Coulomb interaction

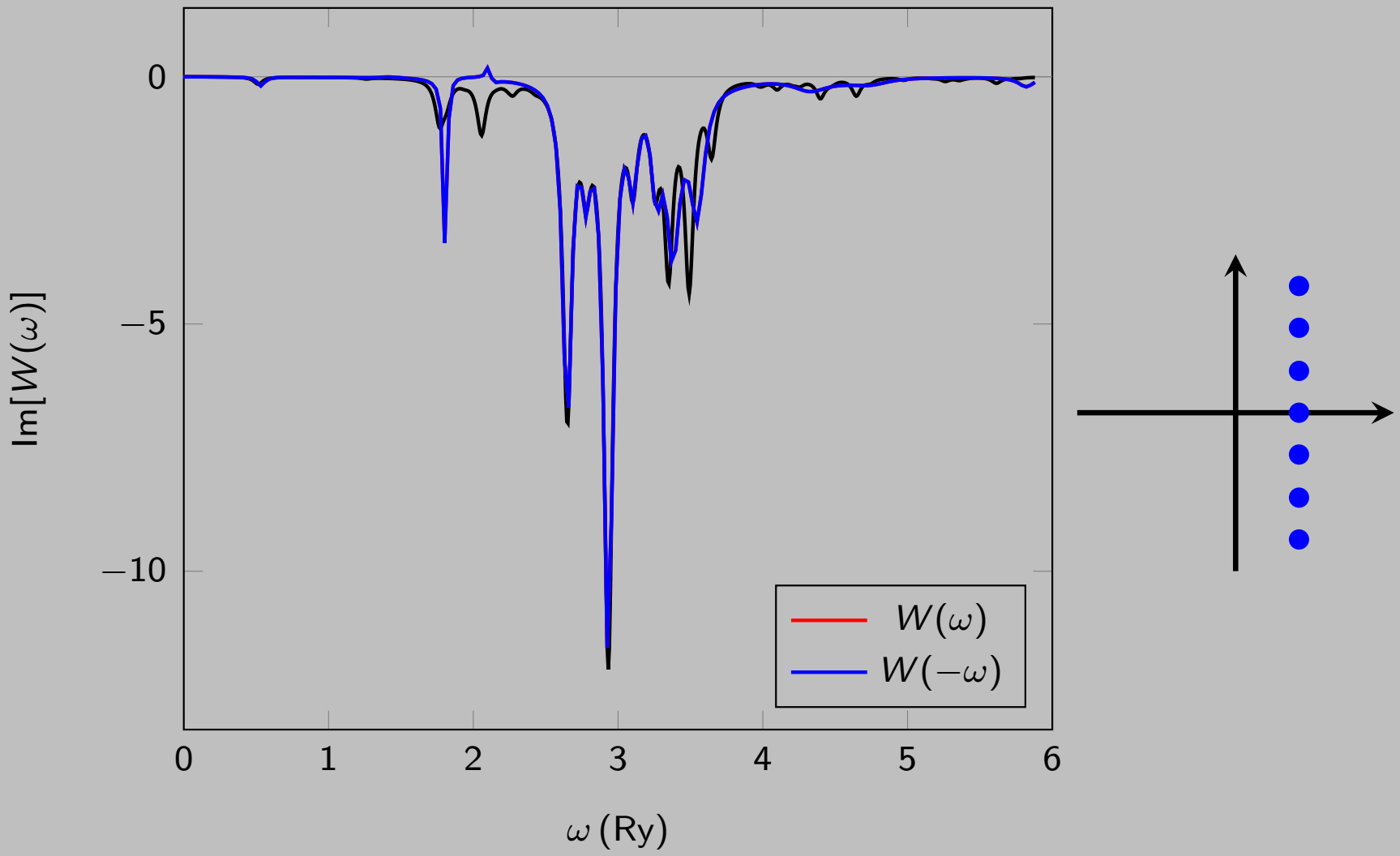
$$W(\omega) = W(-\omega)$$

- conventional Padé approximation

$$W(\omega) = \frac{\sum_m w_m \omega^m}{\sum_n w_n \omega^n}$$

- symmetry fulfilling Padé approximation

$$W(\omega^2) = \frac{\sum_m v_m \omega^{2m}}{\sum_n v_n \omega^{2n}}$$

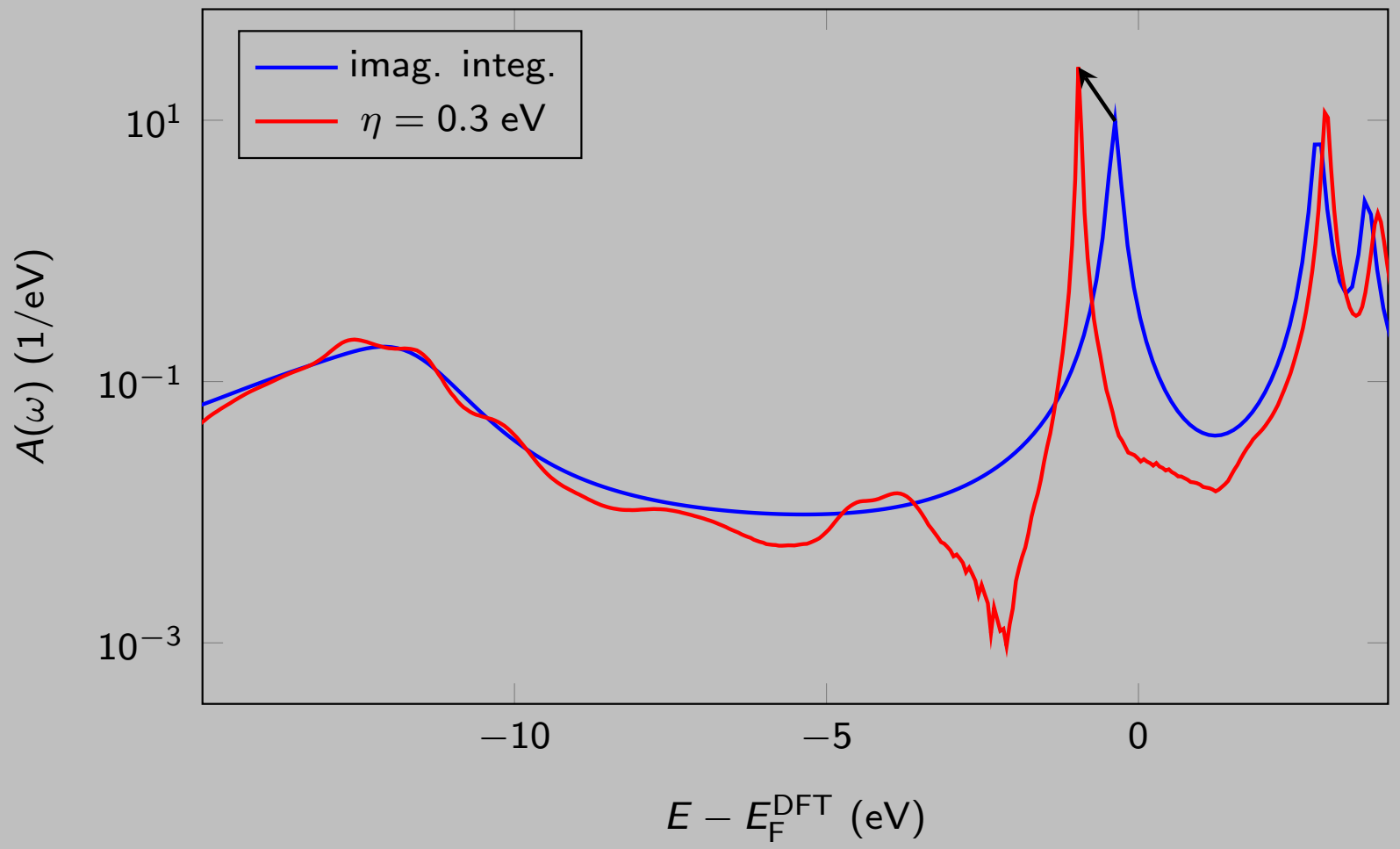

0
−5
−10
Im[W(ω)]
0
1
2
3
4
5
6
ω (Ry)
W(ω)
W(−ω)

Materials Modelling & Design Group

UNIVERSITY OF OXFORD

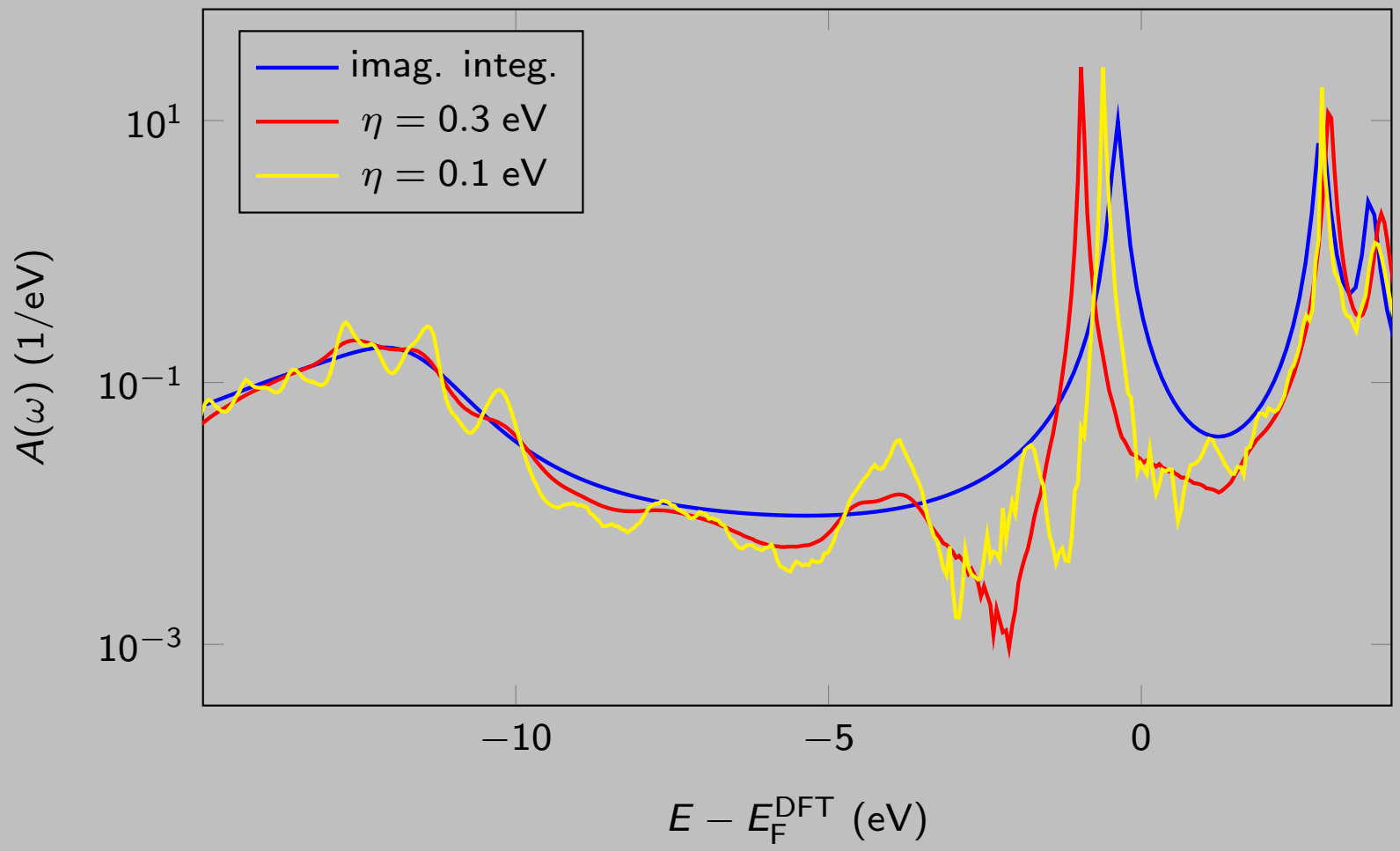
Convolution of G and W

Real vs. imaginary axis integration



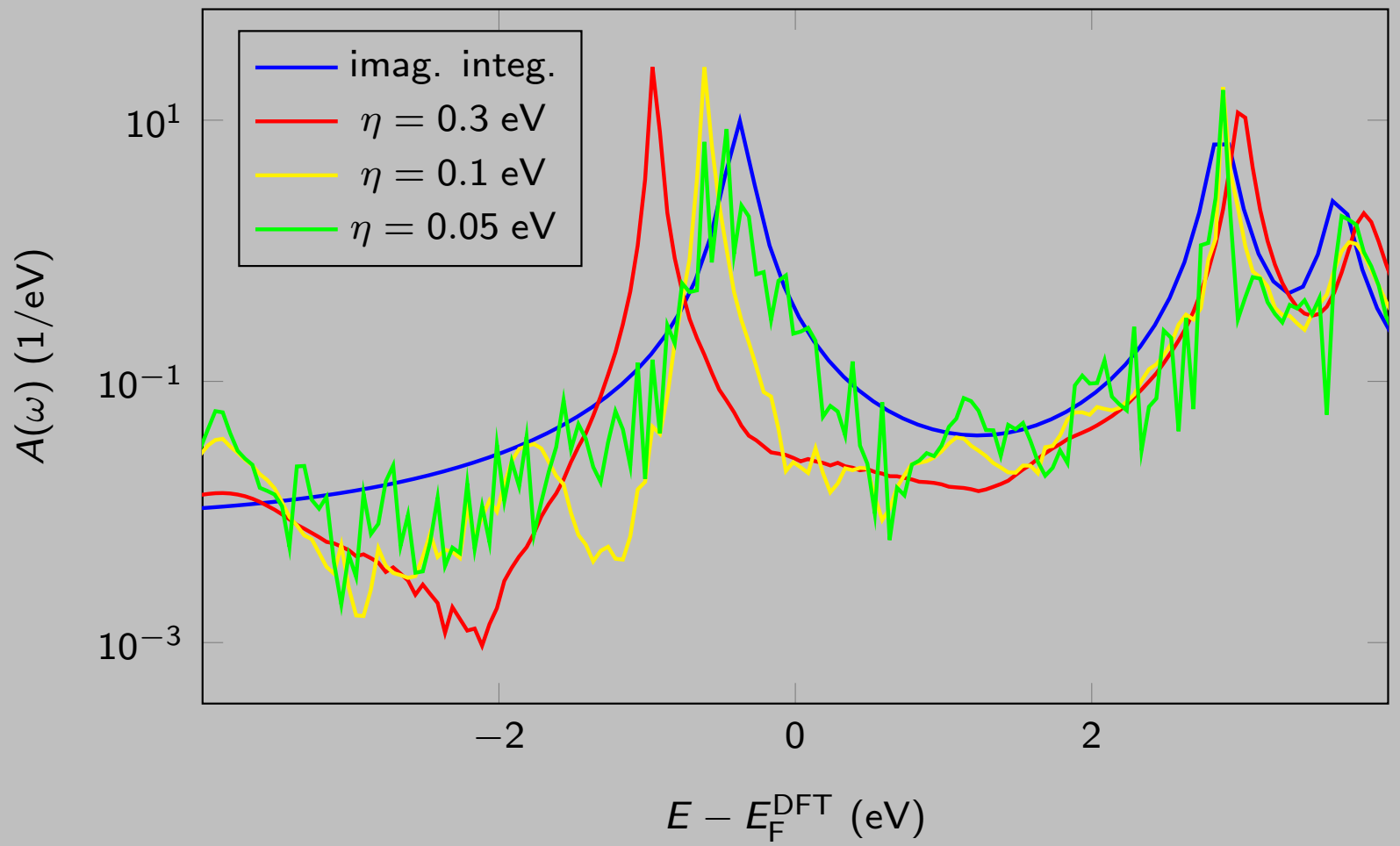
Convolution of G and W

Real vs. imaginary axis integration



Convolution of G and W

Real vs. imaginary axis integration

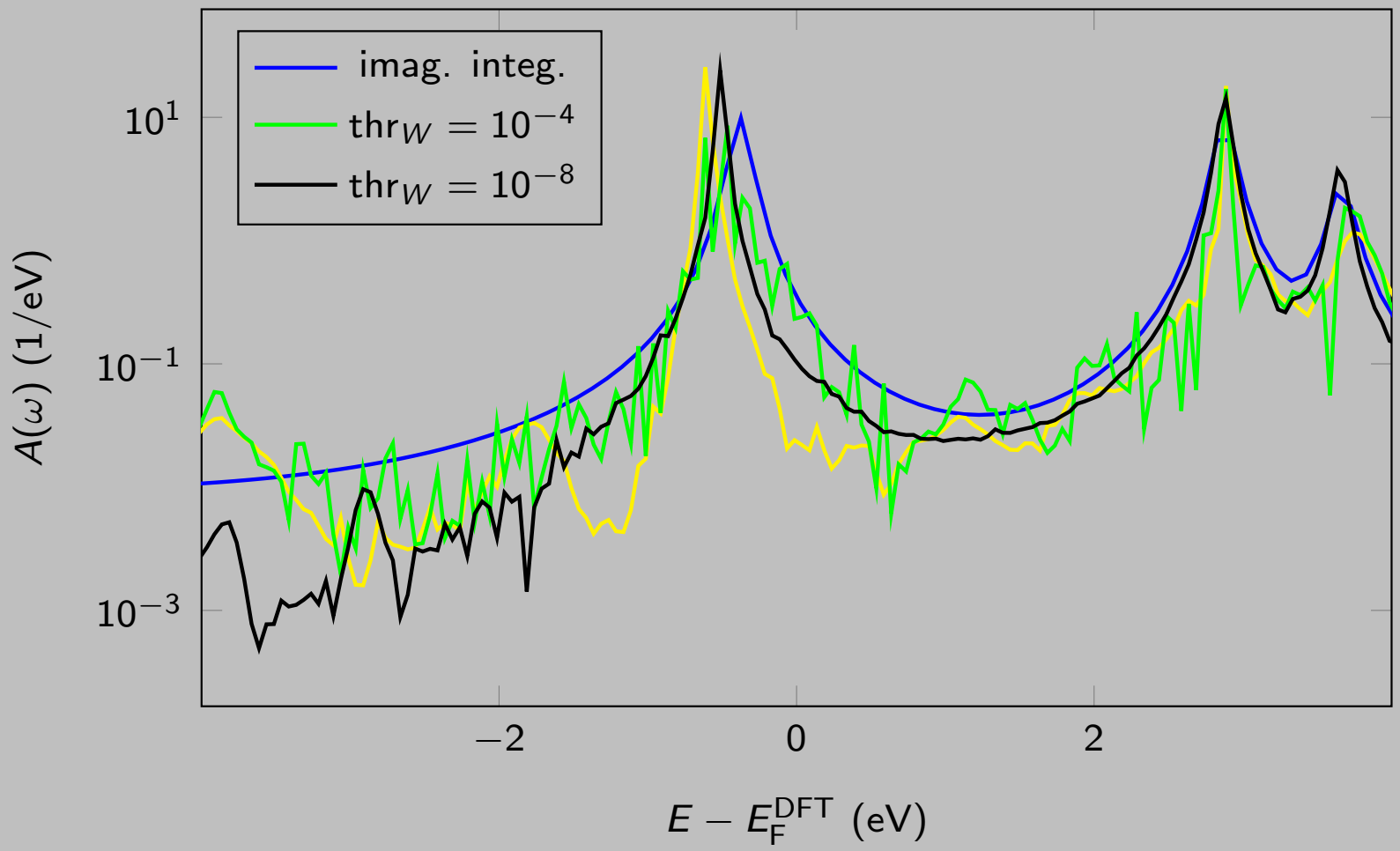


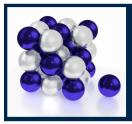
Materials Modelling & Design Group

UNIVERSITY OF OXFORD

Convolution of G and W

Real vs. imaginary axis integration





Previous approach to $6d$ FFT

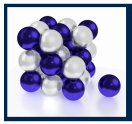
$$G(\boldsymbol{r}, \boldsymbol{r}') \leftrightarrow G(\boldsymbol{G}, \boldsymbol{G}')$$

- Custom FFT grid for both indices (`fft_custom`)
- Outer index parallelized over images
- Wave function use a different FFT grid

New approach for $6d$ FFT

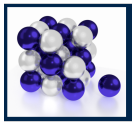
$$G(\boldsymbol{r}, \boldsymbol{r}') \leftrightarrow G(\boldsymbol{G}, \boldsymbol{G}')$$

- Generate custom subgrid of wave function grid (`ggens`)
- Extension to generate `ig_l2g` for parallelized grid
- N.B. `ggens` is not completely generic



Outlook for 2018

- Further reduction of global modules
- Deeper integration into Quantum Espresso
- Postprocessing routines to visualize G, W, and Σ



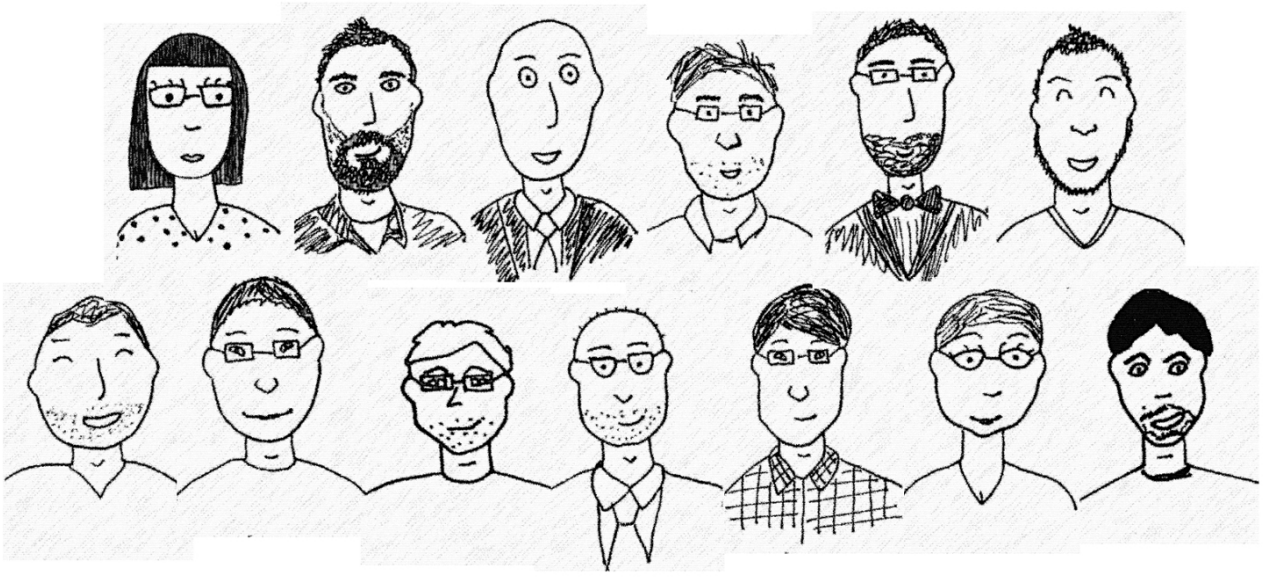
Wishlist for Quantum Espresso

- Gitlab continuous integration
- Developers roadmap
  - quarterly(?) update on planned changes
  - ideally self-organized

Materials Modelling & Design Group

UNIVERSITY OF OXFORD

Acknowledgments



PRACE

arc advanced research computing

The Leverhulme Trust

QUANTUMESPRESSO