Wavelets for everything

Wei Ku (顧 威)
CM-Theory, CMPMSD, Brookhaven National Lab
Department of Physics, SUNY Stony Brook

BROOKHAVEN NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

Acknowledgement

Funding sources

Basic Energy Science, Office of Science, Department of Energy

Former group members

William Garber
BNL

Dmitri Volja
MIT

References

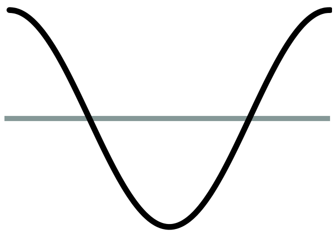
Wavelets

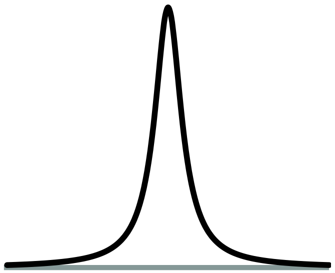
- Daubechies, I., *Ten Lectures on Wavelets*, SIAM, Philadelphia (1992).
- Cohen, A., Daubechies, I. and Feauveau, J.-C., *Biorthogonal Bases of Compactly Supported Wavelets*, Communications on Pure and Applied Math, Vol. XLV, 485-560 (1992).
- Beylkin, G., Keiser, J. M., *An Adaptive Pseudo-Wavelet Approach for Solving Nonlinear PDEs* Wavelet Analysis and Applications, Vol.6, Academic Press (1997).
- Wim Sweldens, *The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets*, Applied and Computational armonic Analysis, 186 200 (1996).

Applications in Electronic Structure calculaiton

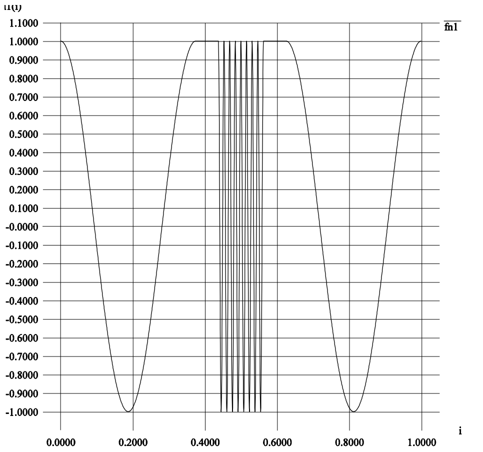
- Harrison, R. J. , Fann, G. I., Yanai, T. , Gan, Z. , Beylkin, G. *Multiresolution quantum chemistry: Basic theory and initial applications*, Journal of Chemical Physics, Vol 121. N. 23, 11587-11598 (2004).
- Sekino, H., Maeda, Y., Yanai, T., Harrison, R., *Basis set limit Hartree-Fock and density functional theory response property evaluation by multiresolution multiwavelet basis*, Journal of Chemical Physics, Vol 129. N. 3, 034111.1-6 (2008).
- Arias, T., *Multiresolution analysis of electronic structure: semicardinal and wavelet bases*, Reviews of Modern Phys., Vol 71, N. 1, 267-311 (1999)

Information and efficiency





vs.



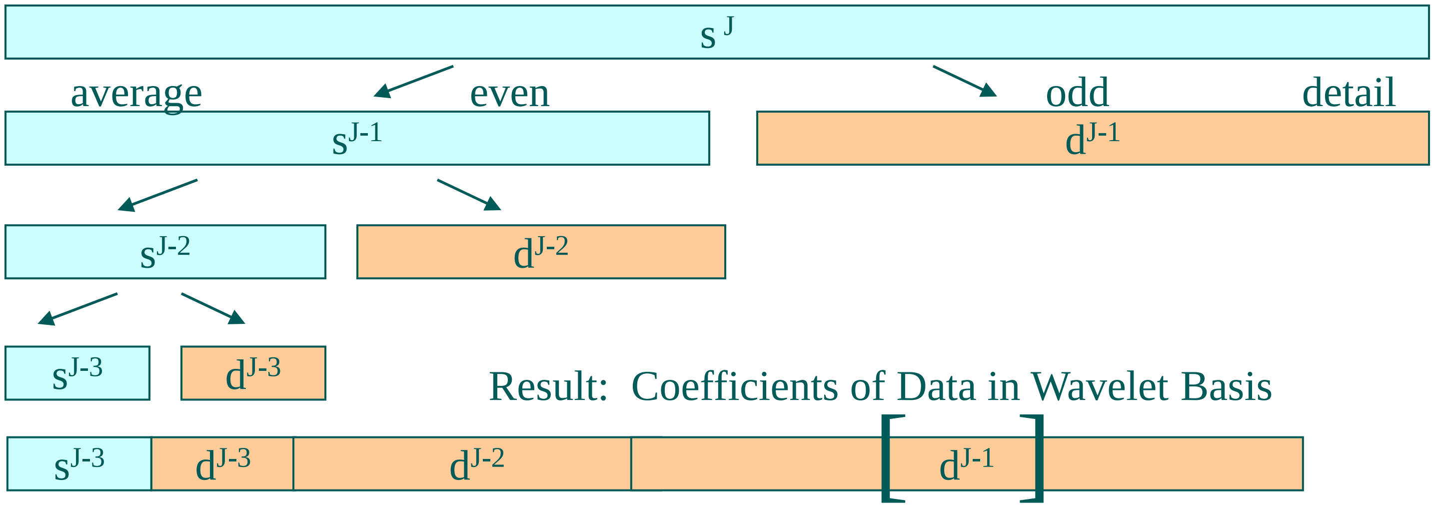
- In the absence of prior knowledge of the structure of the information, an efficient representation needs to be self-adaptive:

→ capturing the smooth average feature and the sharp detail simultaneously.

$$f(x) = \sum_i c_i b_i(x)$$

→ $b_i(x)$ should be compact in both real space and Fourier space

Wavelet transform



$$V_J = V_{j-1} \oplus W_{j-1} = V_0 \oplus W_0 \oplus \ldots \oplus W_{j-1}$$

- compact support → computational efficiency: O(N) faster than FFT O(N log(N))
- position independent transform → same basis function everywhere
- built-in multi-resolution characteristic → same basis function of different width
- s: averaged information, small amount of dense data

  ←→ basis function named "scaling function" $\phi(x)$ spanning V
- d: detailed information, sparse data only near sharp feature

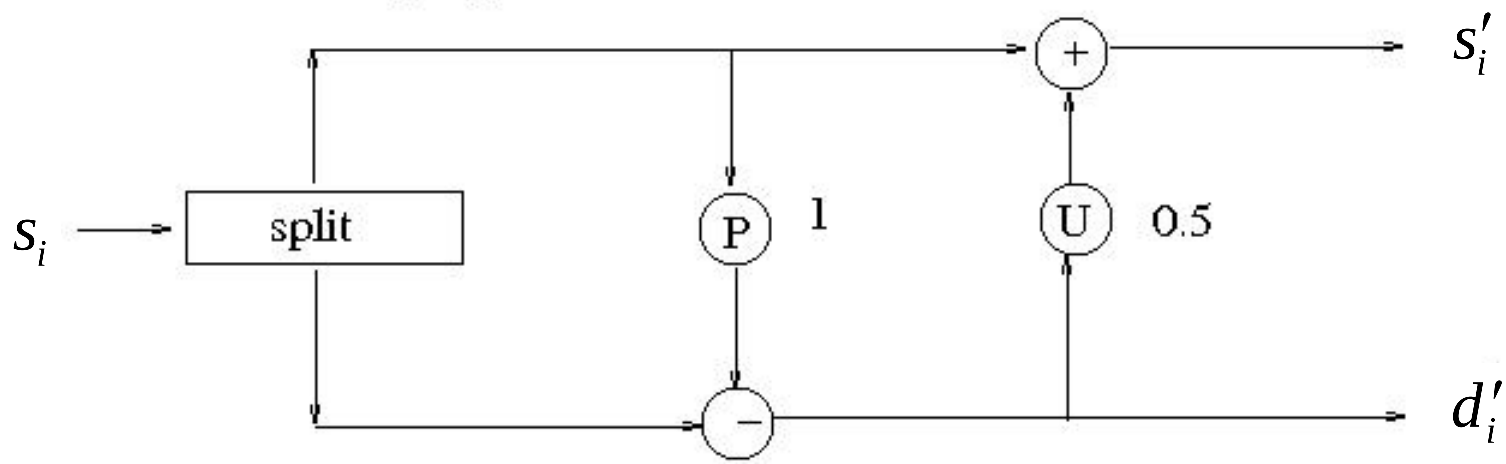
  ←→ basis function named "wavelet" $\psi(x)$ spanning W

Lifting algorithm as an example

- 1-step in CDF(2,2) wavelet transform
  Cohen, Daubechies, and Feauveau

$$s_i' = s_{2i}$$

$$s_i' = s_i' + \left(d_{i-1}' + d_i'\right)/2$$
$$= \left(-s_{2i-2} + 2s_{2i-1} + 6s_{2i} + 2s_{2i+1} - s_{2i+2}\right)/8$$



$$d_i' = s_{2i+1}$$

$$d_i' = d_i' - \left(s_i' + s_{i+1}'\right)/2$$
$$= \left(-s_{2i} + 2s_{2i+1} - s_{2i+2}\right)/2$$

- inverse transform → reverse the operation

Wavelet transform

Repeat the two-scale relation until the coarsest level is reached

FWD $$s_k^j = \sum_m \tilde{h}_{m-2k}\, s_m^{j+1}$$

$$d_k^j = \sum_m \tilde{g}_{m-2k}\, s_m^{j+1}$$

INV $$s_m^{j+1} = \sum_k \left(h_{m-2k}\, s_k^j + g_{m-2k}\, d_k^j \right)$$

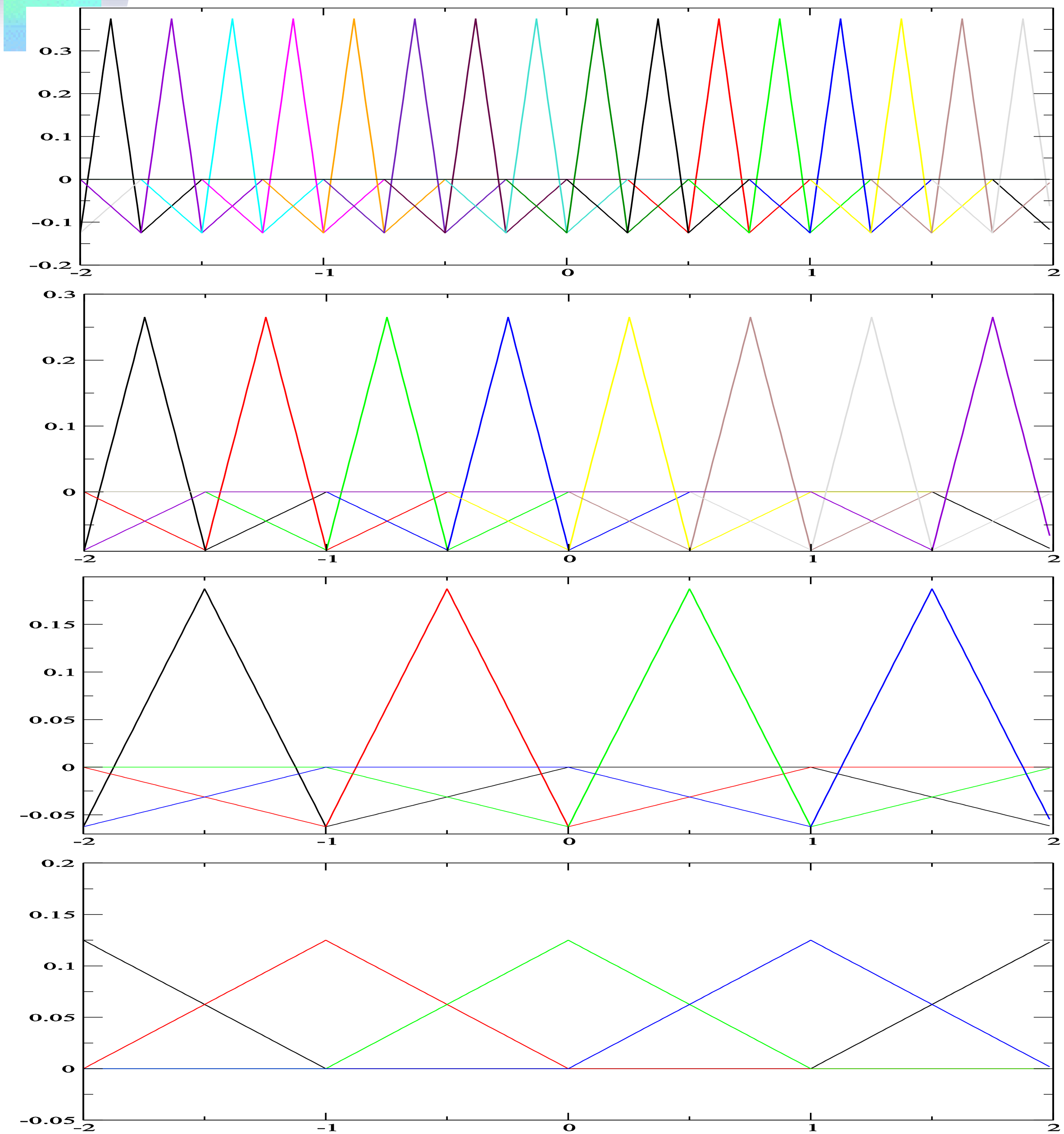
where

$$s_k^j = \left\langle \tilde{\varphi}_{j,k} \middle| f \right\rangle \qquad d_k^j = \left\langle \tilde{\psi}_{j,k} \middle| f \right\rangle$$

$$\tilde{h}_m = \left\langle \tilde{\varphi}_{j,k} \middle| \varphi_{j+1,m+2k} \right\rangle \quad \tilde{g}_m = \left\langle \tilde{\psi}_{j,k} \middle| \varphi_{j+1,m+2k} \right\rangle$$

$$h_m = \left\langle \tilde{\varphi}_{j+1,m+2k} \middle| \varphi_{j,k} \right\rangle \quad g_m = \left\langle \tilde{\varphi}_{j+1,m+2k} \middle| \psi_{j,k} \right\rangle$$

CDF(2,2) wavelets



W_2 fine

$\oplus$ $\downarrow$

W_1 wavelets

$\oplus$ $\downarrow$

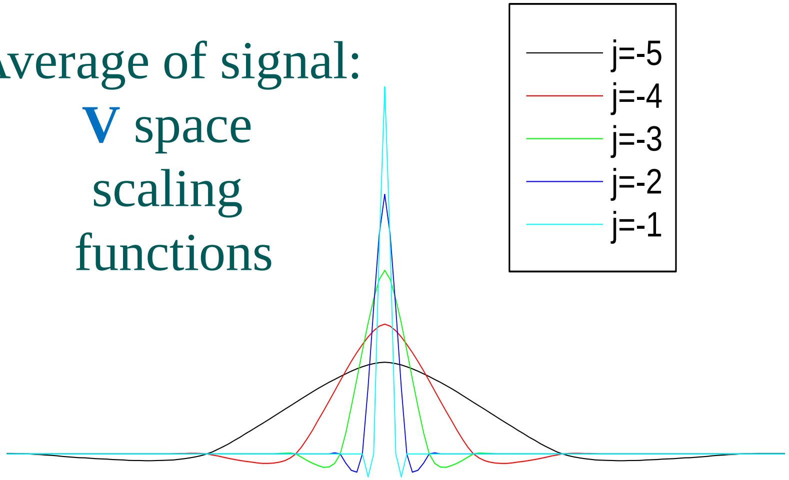
W_0

$\oplus$ $\downarrow$

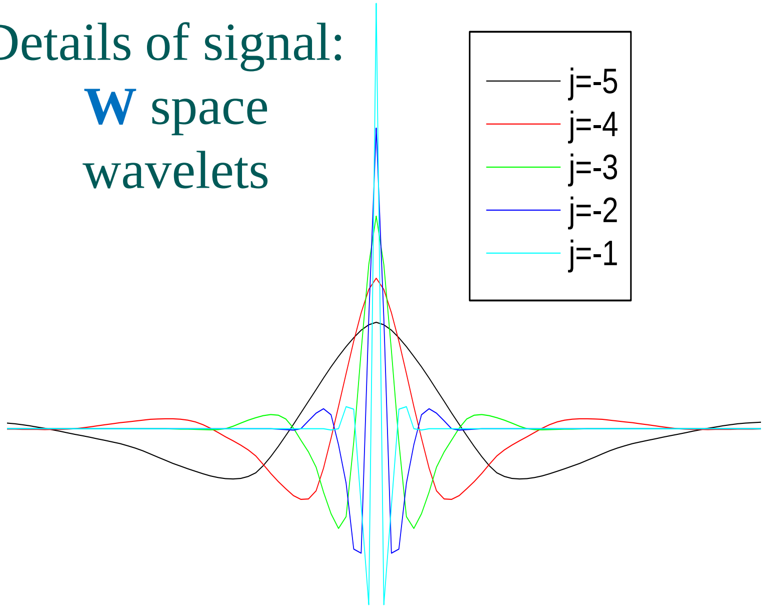
V_0 coarse scaling functions

CDF(4,4) wavelets

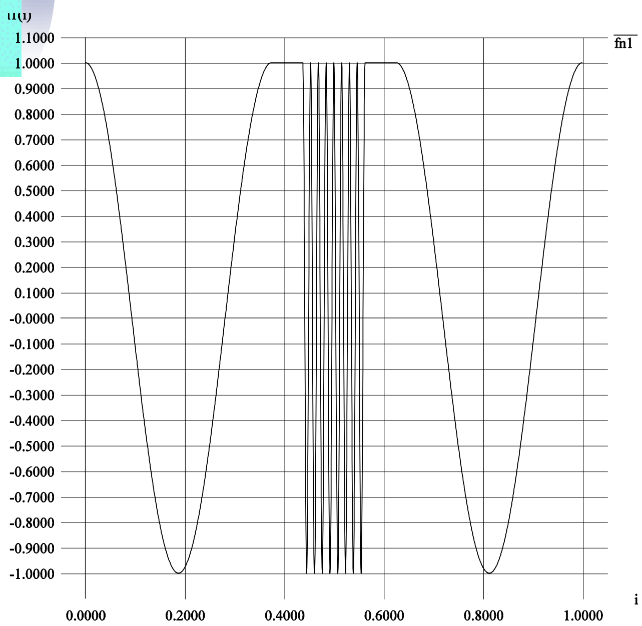
Average of signal:
V space
scaling
functions



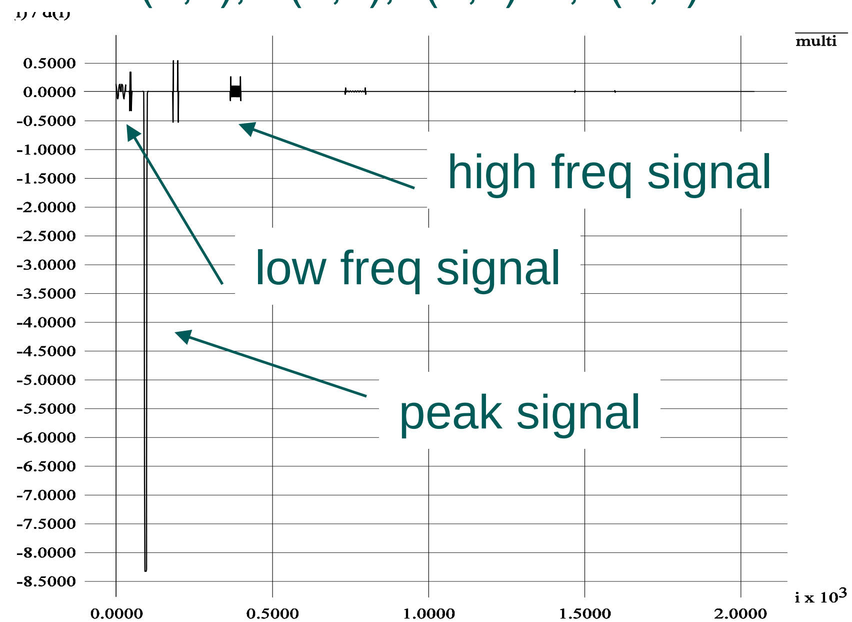
Details of signal:
W space
wavelets



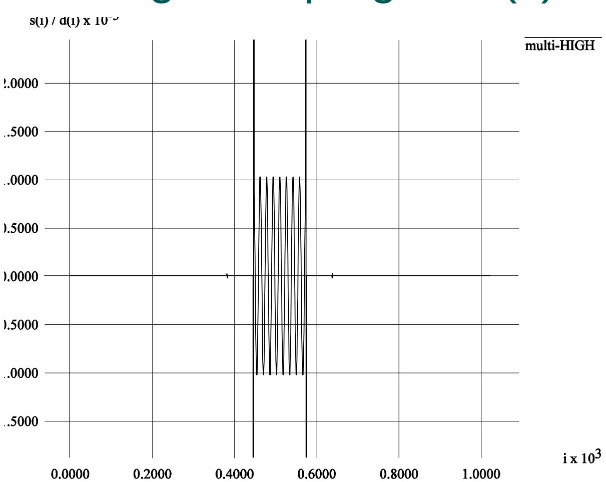
Sampled Function



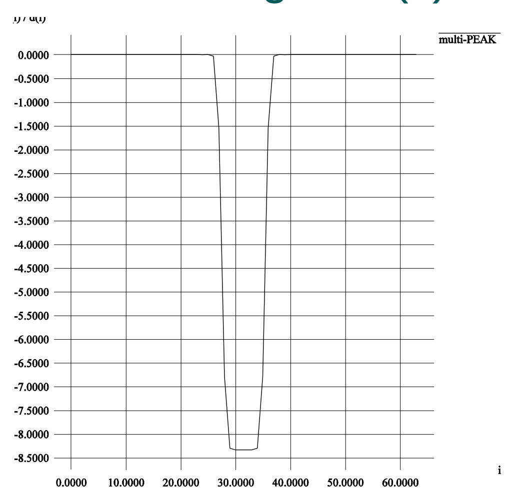
Transformed data s(0,k), d(0,k),d(1,k)…,d(J,k)



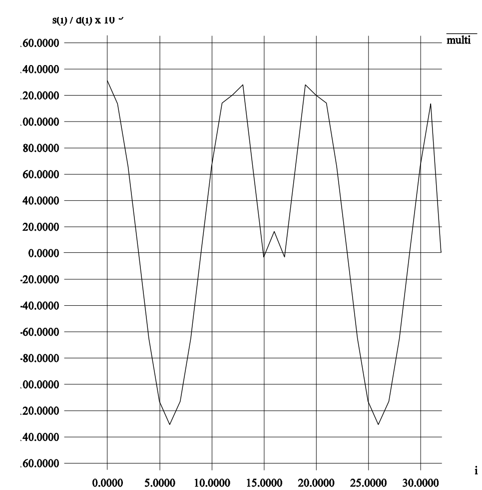
High Freq Signal d(k)



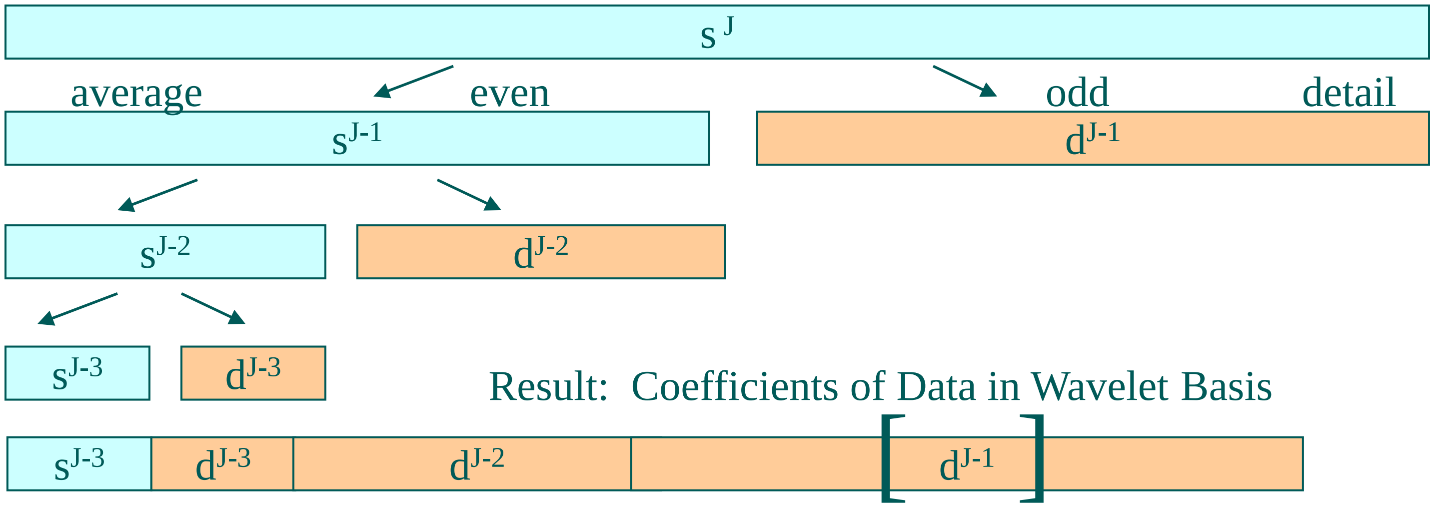
Peak Signal d(k)



Low Freq Signal s(k)



Wavelet transform



$$V_J = V_{j-1} \oplus W_{j-1} = V_0 \oplus W_0 \oplus \ldots \oplus W_{j-1}$$

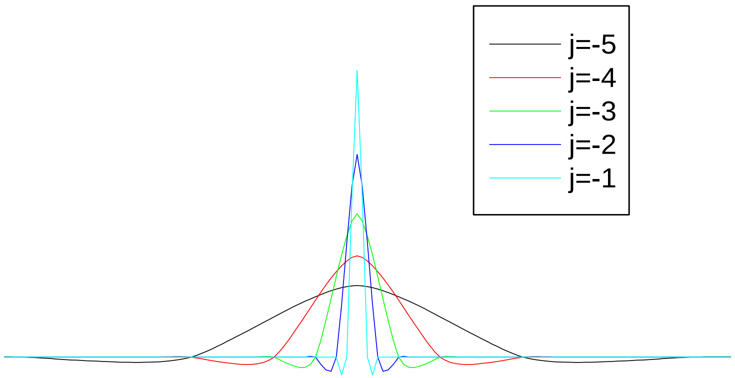
- compact support → computational efficiency: O(N) faster than FFT O(N log(N))
- position independent transform → same basis function everywhere
- built-in multi-resolution characteristic → same basis function of different width
- *s*: averaged information, small amount of dense data

  ←→ basis function named "scaling function" $\varphi(x)$ spanning V
- *d*: detailed information, sparse data only near sharp feature

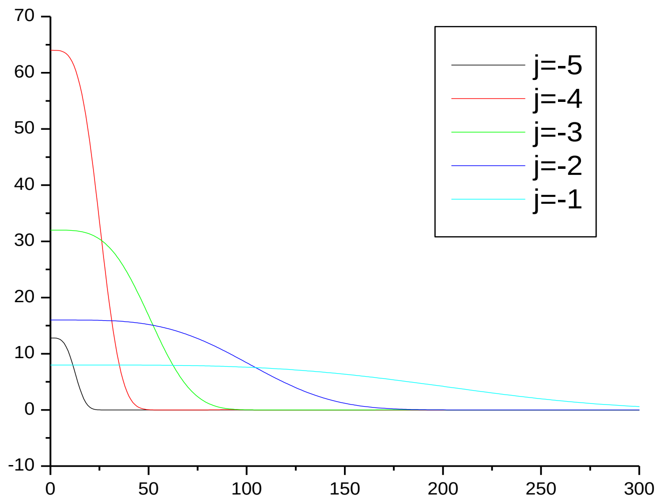
  ←→ basis function named "wavelet" $\psi(x)$ spanning W

Properties of wavelets

- Strict compactness in real space



- Compactness in Fourier space

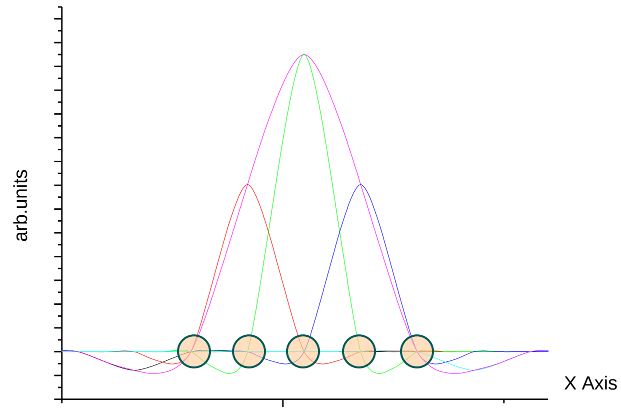


- Bi-orthogonal: duals also wavelets

  → overlap matrix unnecessary

- CDF (4,4) wavelets: Interpolating

  $f(x_i) = c_i$ for φ_i centered at x_i



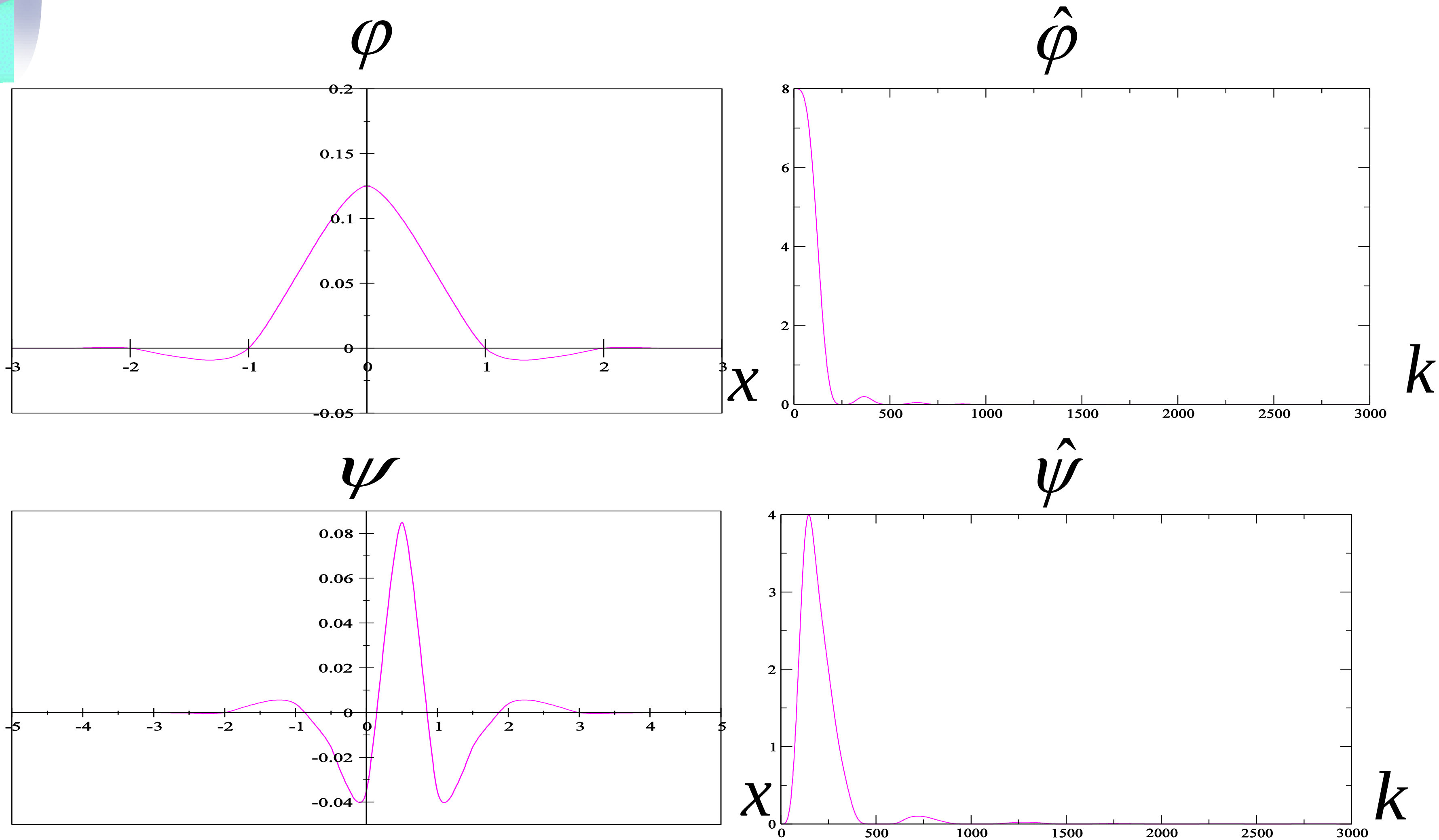
- vanishing low-order moments

$$\int \varphi(x)\,dx = 1; \quad \int x^n \varphi(x)\,dx = 0; \quad n = 1 \ldots N-1$$

$$\int x^n \psi(x)\,dx = 0; \quad n = 0 \ldots N-1$$

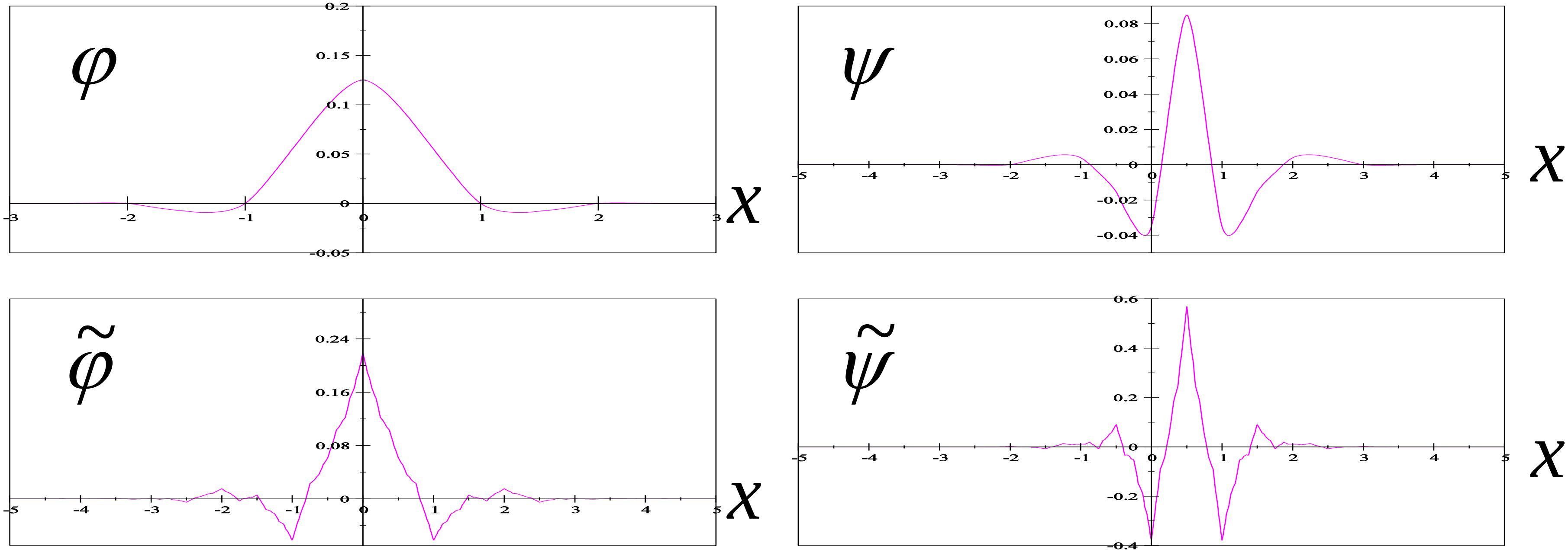
- fits 4th order polynomials
- multipole expansion → monopole

Compact in both real and Fourier space



- compact support in x } simultaneously
- localized in k.
- Potential V(x) sparse
- Kinetic Energy T(k) sparse

Bi-orthogonality



completeness: $$\mathbf{1} = \sum_k \left|\varphi_{0,k}\right\rangle\left\langle\tilde{\varphi}_{0,k}\right| + \sum_{j,k}\left|\psi_{j,k}\right\rangle\left\langle\tilde{\psi}_{j,k}\right|$$

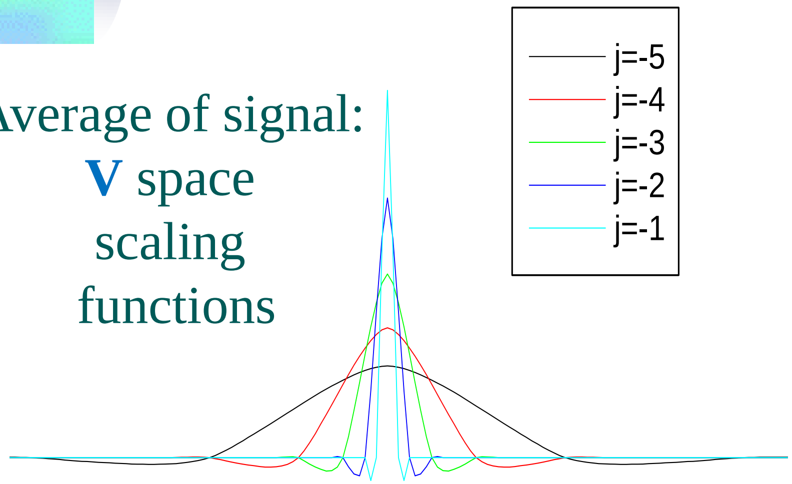
biorthogonality: $$\left\langle\tilde{\varphi}_{0,k}\middle|\varphi_{0,k'}\right\rangle = \delta_{k,k'} \quad \left\langle\tilde{\psi}_{j,k}\middle|\varphi_{0,k'}\right\rangle = 0$$

$$\left\langle\tilde{\varphi}_{0,k}\middle|\psi_{j,k'}\right\rangle = 0 \quad \left\langle\tilde{\psi}_{j,k}\middle|\psi_{j,k'}\right\rangle = \delta_{j,j'}\delta_{k,k'}$$

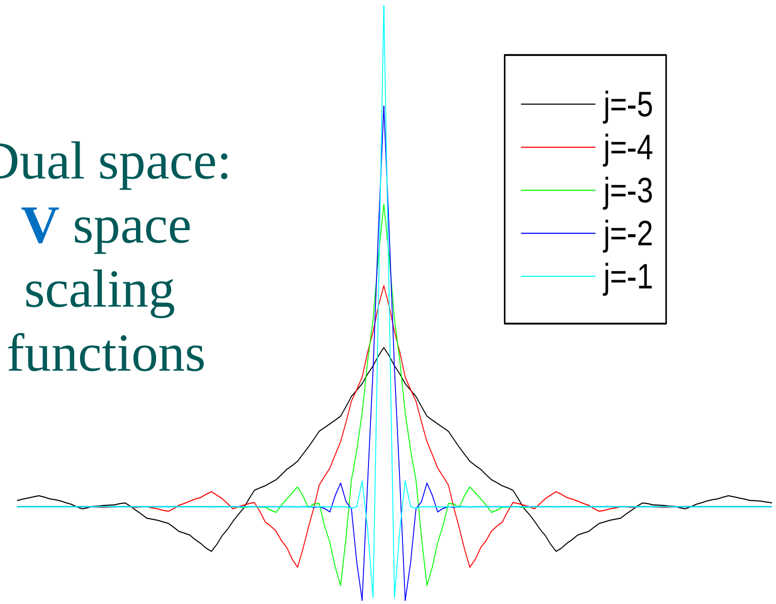
Duals of wavelets are also wavelets.

Bi-orthogonality

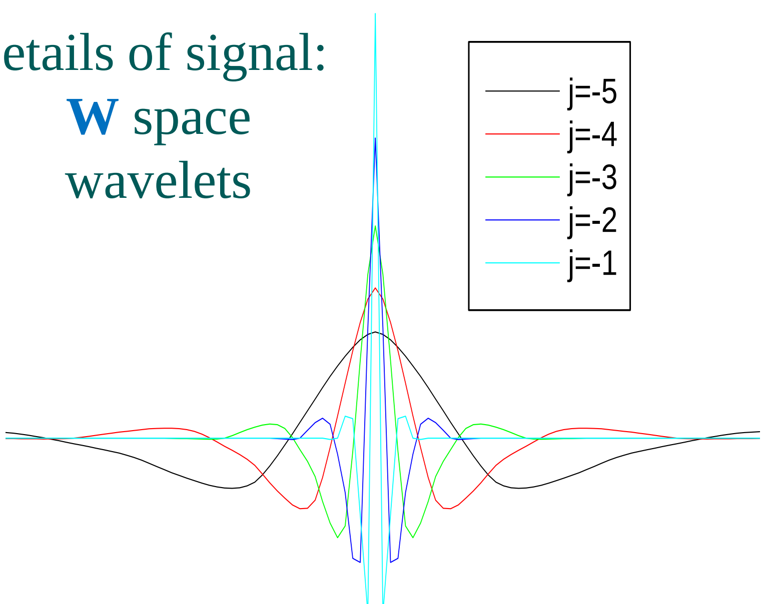
Average of signal:
V space
scaling
functions



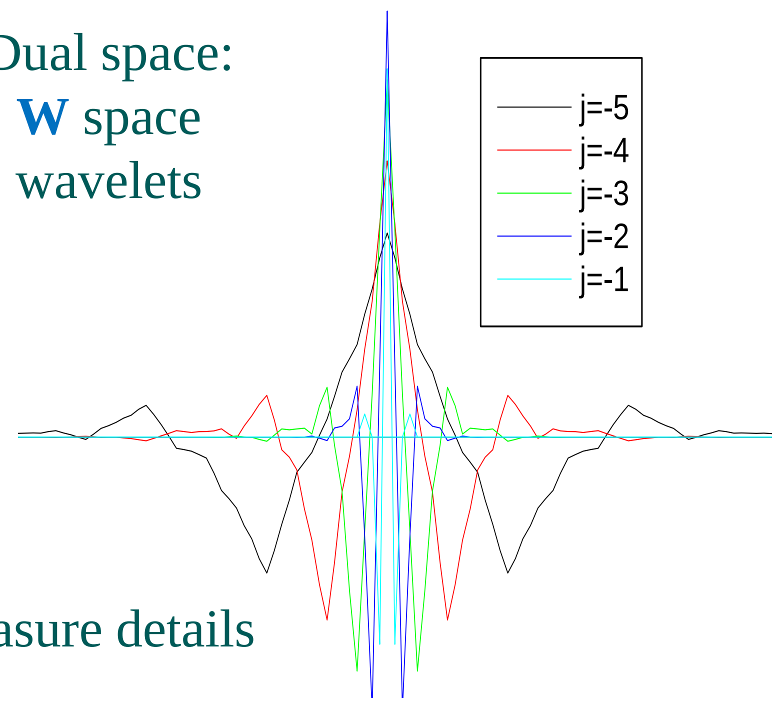
Dual space:
V space
scaling
functions



Details of signal:
W space
wavelets

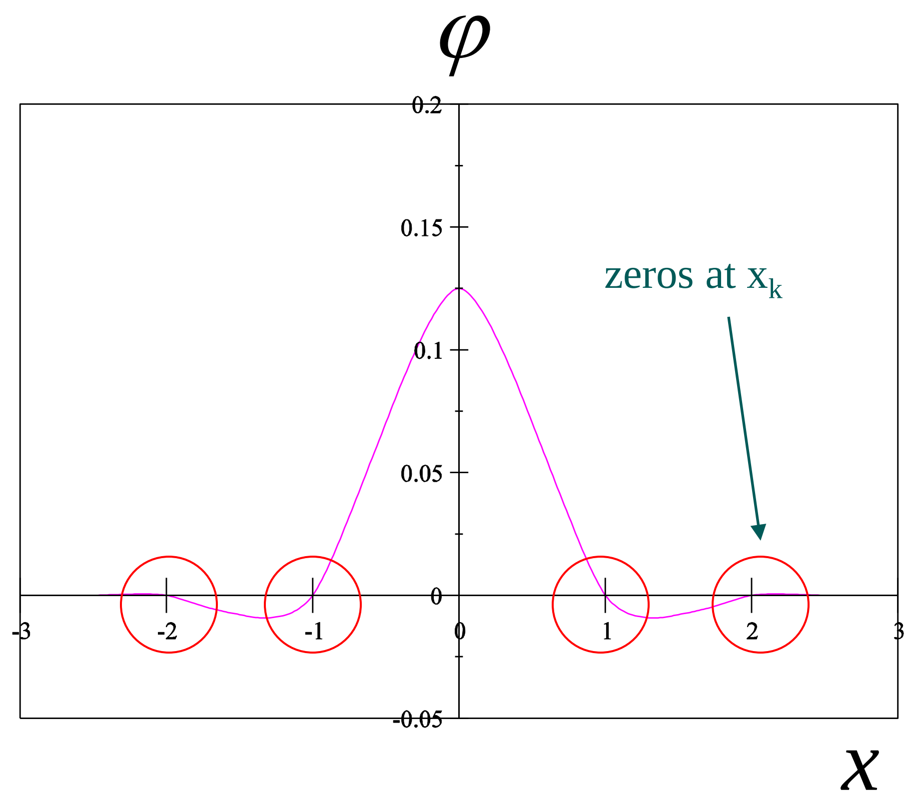


Dual space:
W space
wavelets



all wavelets have zero average: coeffs measure details

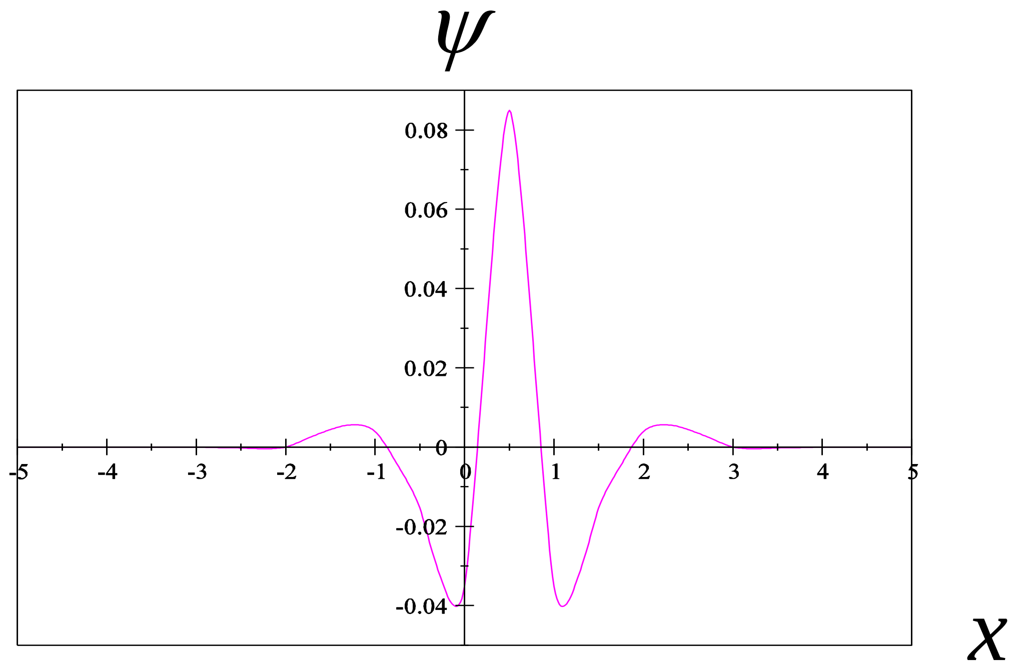
CDF(*N*,*N'*) interpolating wavelets



φ has zeros at x_k so
coefficienk s_k equals value of function at grid points at this scale.

$$f(x) = \sum_k s_k \varphi_k(x) \qquad \varphi_{k_1}(x_{k_2}) = \delta_{k_1,k_2} \qquad f(x_k) = s_k$$

Moment conservation of CDF(*N*,*N'*) wavelets



DD(4,4) wavelet ψ

- interpolates polynomials up to x^3
- has zero moments up to 3rd order.

Coulomb interaction trivial and efficient.

- Only 0^{th} moment of φ contributes.
- Acts like small number of point charges.

$$\int \psi\, dx = 0 \quad \int x\psi\, dx = 0 \quad \int x^2\psi\, dx = 0 \quad \int x^3\psi\, dx = 0$$

$$\boxed{\int \varphi\, dx = 1} \quad \int x\varphi\, dx = 0 \quad \int x^2\varphi\, dx = 0 \quad \int x^3\varphi\, dx = 0$$

Example: significant reduction of multipole expansion

$$\Phi(x) = \int \frac{\rho(x')}{|x - x'|} d^3x'$$

$$\Phi(x) = \frac{q}{r} + \frac{p \bullet x}{r^3} + \frac{1}{2}\sum_{i,j} Q_{i,j} \frac{x_i x_j}{r^5} + \text{higher} - \text{order}$$

$$Q_{i,j} = \int \left(3x_i' x_j' - r'^2 \delta_{i,j}\right) \rho(x') d^3x'$$

for ρ represented by CDF44 wavelets,

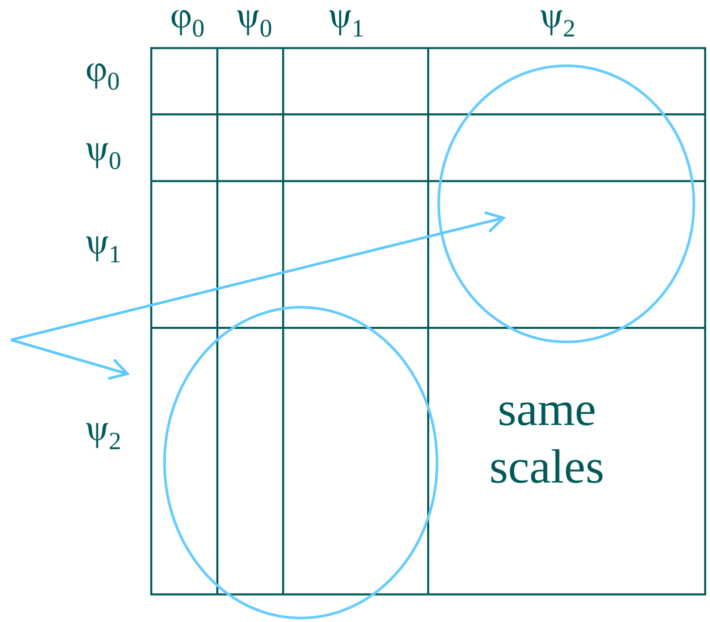
first 3 moments are zero, so

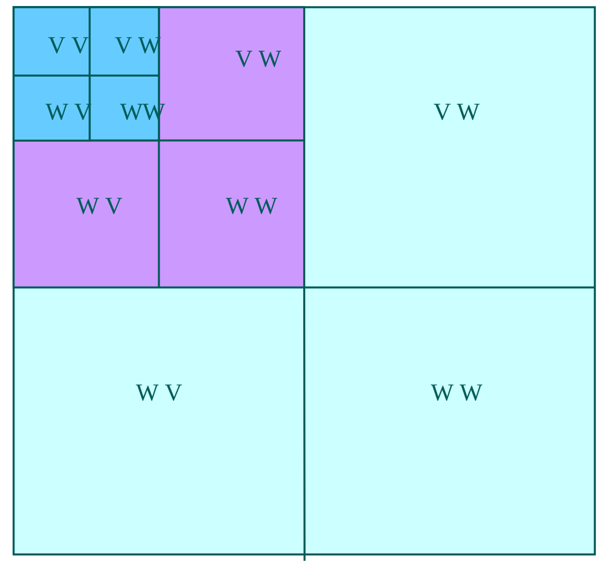
$q,\ p,\ Q_{i,j}$ are computed from the coarse scale data only :

(scaling function coefficients)

There is much less data to compute.

Higher dimension: tensor wavelets in nonstandard form





separate scales treated separately;

no mixed scales

Standard Form:
Forward Transform X and Y
Recur on whole row/col
Disadvantage:
mix scales; Operator matrix *not simple*

Nonstandard Form:
Forward Transform X and Y
Recur on V V average data
Advantage:
Data sparse; Operator matrix sparse

Operator Matrix (Laplacian):
recur on V V block. Do not mix scales
COMPACT SUPPORT → O(N): within each scale, matrices are banded
All operations O(N)

$$\text{Block of Matrix} = \left\langle \psi_{j,k1} \middle| \nabla^2 \middle| \psi_{j,k2} \right\rangle = \left\langle \psi_{j,k1-k2} \middle| \nabla^2 \middle| \psi_{j,0} \right\rangle$$

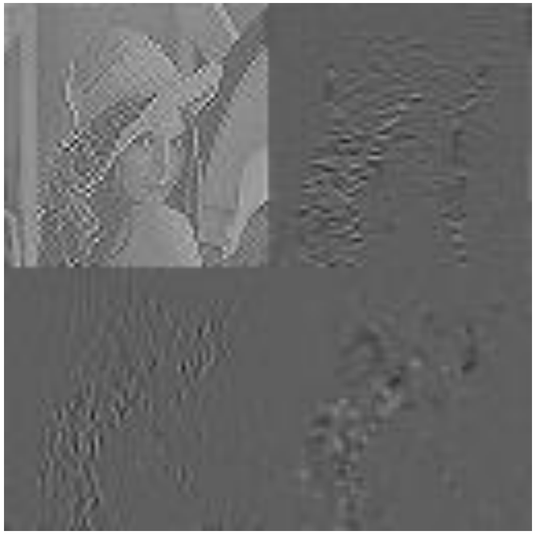
Example: 2D cubic spline forward transform

Original 512x512

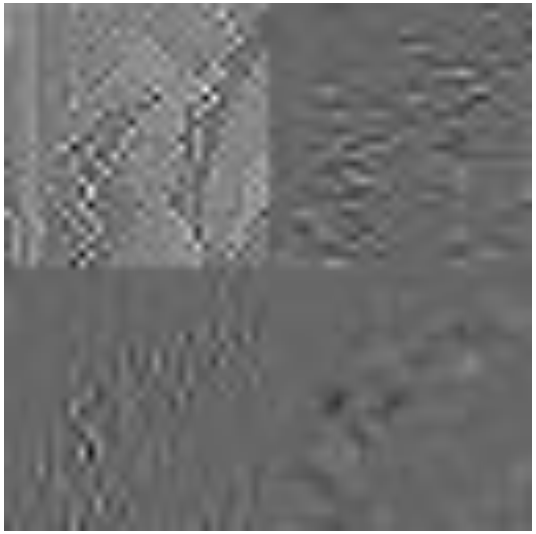
Level 4 256x256

Level 3 128x128

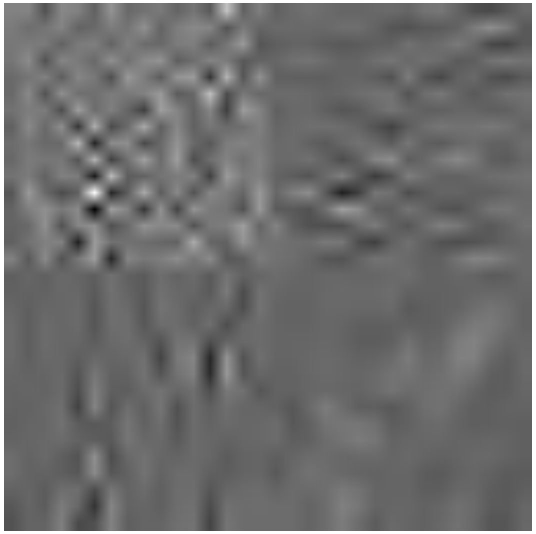
Level 2 64x64



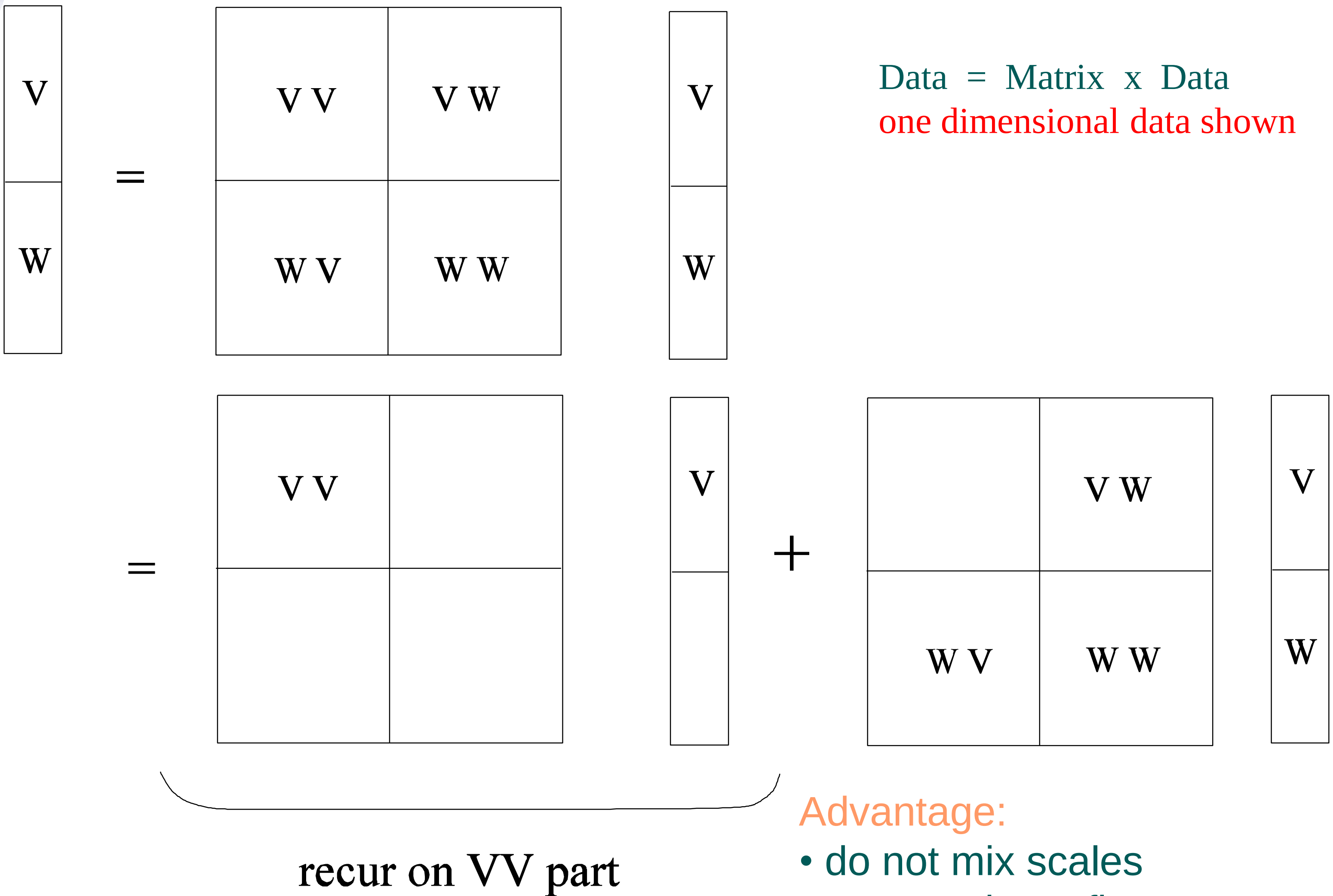
Level 1 32x32



Level 0 16x16



Linear algebra: matrix vector multiplication



Data = Matrix x Data

one dimensional data shown

Advantage:

- do not mix scales
- progressive refinement
- for translationally invariant matrix,
  blocks are simple filters: O(N)

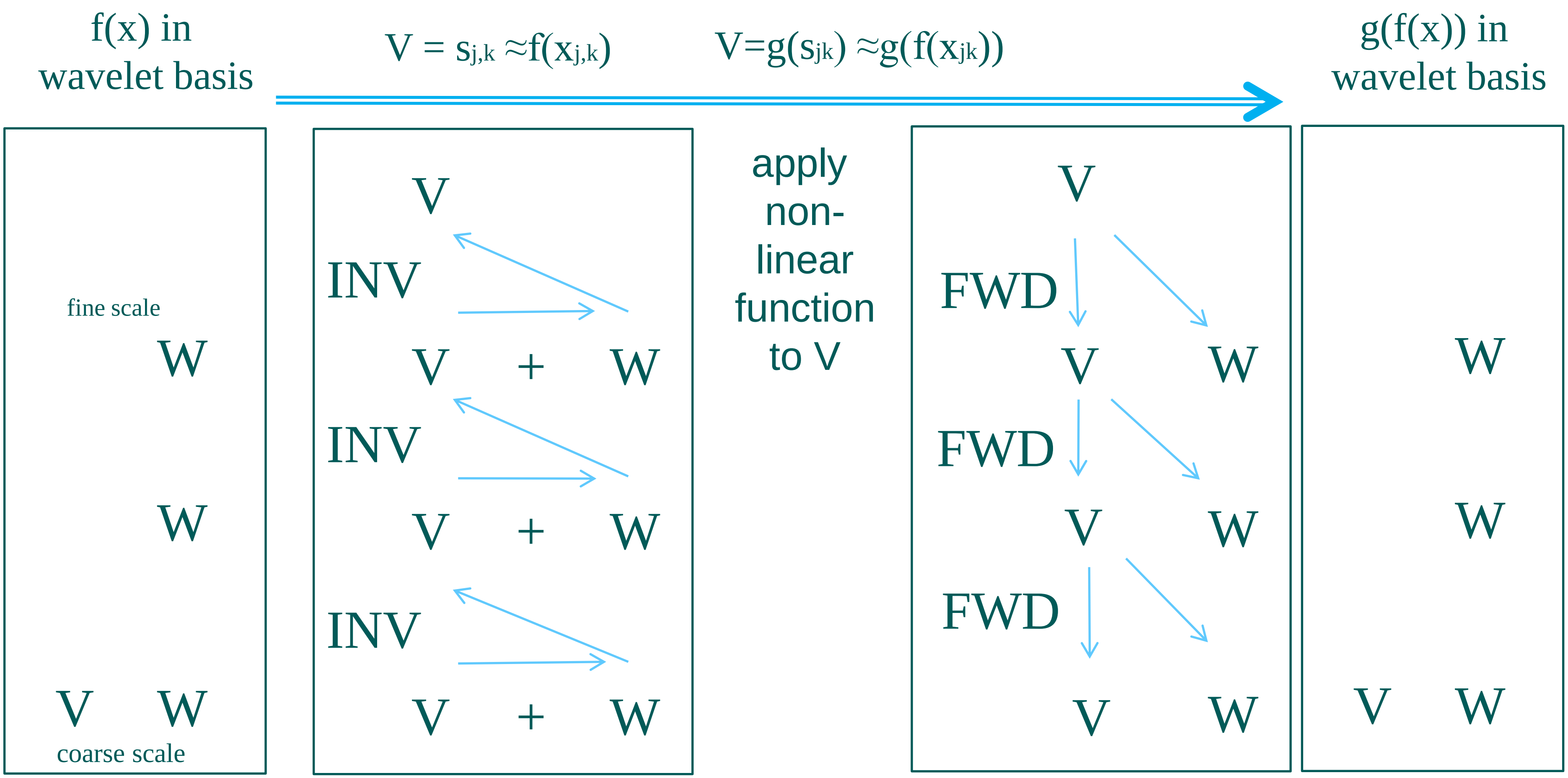
Timing: Laplacian operator

j	Size, m x m	Time, sec		**Speed = m^2/T (10^6/s)**		**Speed, Sparse/ Dense**
		Dense	Sparse	**Dense**	**Sparse**	
2	1024x1024	2.9	1.4	0.36	0.75	2.1 x faster
3	2048x2048	11.5	4.1	0.36	1.02	2.8 x faster
4	4096x4096	83	21.5	0.20	0.78	3.9 x faster
5	8192x8192	837 (swaps)	98	0.080	0.68	8.5 x faster

- Sparse wavelets faster than dense
- Handles larger problem with same amount of memory

Non-linear algebra with interpolating wavelets

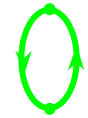
Example : $f = \frac{1}{r}\psi$; $E_{ion} = \langle \psi | f \rangle$



- interpolating property: average data V ≈ value of function at grid points
- remain within sparse representation
- wavelet transform: COMPACT SUPPORT → O(N)

An example for many-body perturbation theory

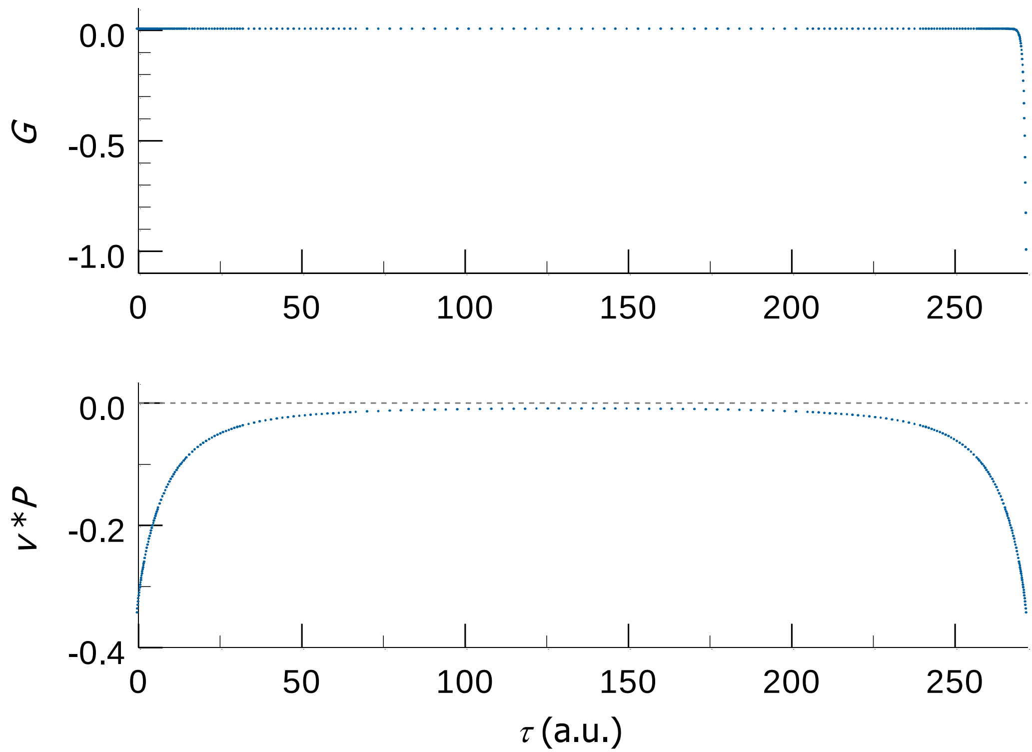
- convolution involving $1/\omega$ tail of $G(\omega)$:



$$P(1,2) = G(1,2) \cdot G(2,1)$$

$$P(\omega_n) = \sum_{i=0}^{\infty} G(\omega_n) \cdot G(\omega_n + \omega_i)$$

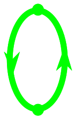
$$P(\tau) = G(\tau) \cdot G(-\tau)$$



- explicit inclusion of $\tau = 0^+$ & 0^- (2^{nd} generation of wavelets)

Non-uniform grid in Matsubara time

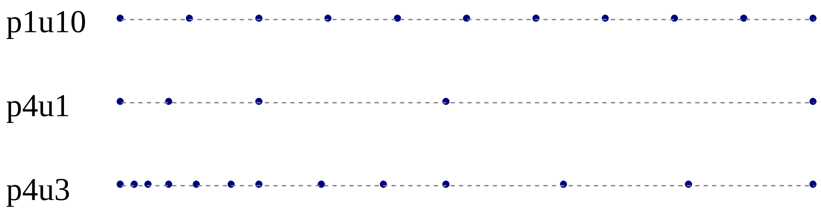
- convolution involving $1/\omega$ tail of $G(\omega)$:



$$P(1,2) = G(1,2)\cdot G(2,1)$$

$$P(\omega_n) = \sum_{i=0}^{\infty} G(\omega_n)\cdot G(\omega_n + \omega_i)$$

$$P(\tau) = G(\tau)\cdot G(-\tau)$$



- This is the same as using the scaling function across level as basis
- Easy to handle mismatched grid point (inverse wavelet transform)

$$W(\tau) = v\cdot\delta(\tau) + \int_0^{\beta} v\cdot P(\tau-\tau')W(\tau')d\tau'$$

Wavelet++ package

Why Use Wavelets:

- compact support in space x
- localized in scale k:
  - high res detail, low res averages
  - systematic control of error
- sparse representation:
  - identify, compute with, store only critical data
  - All operations done without leaving sparse represent.
- conservation of moments
- interpolating properties
- fast O(N) algorithms for
  - wavelet transform
  - differential operators (Laplacian; Kinetic Energy)
  - nonlinear operations (External Potential)
  - products

Applications:

- physical problems
- biorthogonal bases (bra/ket)
- large data sets
- high resolution

Wavelet library

Data Structures:

- Filter
  basic convolution
- LiftingStep
  WaveletDef:
  define wavelet coeff h,g
  provide transform
- WaveletRepDense
  WaveletRepSparse
  store data

Operations:

- forward transform
- inverse transform
- function composition
- product
- convert to dense
- convert to sparse

Vector Space library

Data Structures:

- VectorSpaceDense
  VectorSpaceSparse
  Overlap Matrix for finding Duals
  Explicit treatment of crystal
  translational symmetry
- Bivector
  Wrapper associating
  WaveletRep with VectorSpace
- TranslationalyInvariantMatrix

Operations:

- Inherit Wavelet operations
- DualConj
- AddMult: Matrix Multiply
- InnerProduct

Wavelet++ library is easy to use:
Example of 2D cubic spline forward transform

```
WDEF wav = &cubic_spline;
BASIS basis(wav);
TinyI extent(512,512);
WREP wrep(extent, basis);

loadPhoto(wrep, fnamePhotoIn);
while(nlev-- > 0) {
  string fname = "photo”; fname += nlev + ".dat";
  wrep.transFwd(1);
  savePhoto(wrep, fnamePhoto);
}
```

Wavelet++ library is flexible: define your own class of wavelets

```
typedef WaveletDef<double>          WDEF;
typedef WaveletDefLiftStep<double> LSTEP;

// Haar Wavelet with Lifting Steps
WDEF haar("haar", 1/sq2, sq2,
       LSTEP(LS_PREDICT, 1, 1, -1.0),
       LSTEP(LS_UPDATE,  0, 1, 0.5));

// Daubechies Wavelet as Convolution
h = (1+sq3)*sq2/8,
    (3+sq3)*sq2/8, // Filter coefficients
    (3-sq3)*sq2/8,
    (1-sq3)*sq2/8;
g = h(3), -h(2), h(1), -h(0);
std::vector<LSTEP> v;
v[0] = LSTEP(h,g,h,g);  // convolution step
WDEF daubechies("daubechies", 1, 1, v);
```

Vector space library is easy to use: algebra & interface

dense or sparse:

```
ip = InnerProduct(v1, v2);
ip = InnerProductShift(v1, v2, deltaCell);
vz = AddMult(vy, LaplacianMatrix, vx);
vz = DualConj(vy, vx);
vz = Product(v1, v2, v3);
vz.FunctionComp(vx, functionToApply);
```

summary: using blitz++ algebra on blitz::Array base class

```
v1 += v2 + Product(v3, v4, v5)
         + InnerProduct(v3,v4) * v5
         + AddMult(v6, mat, v7);
```

Vector space library is easy to use: Laplacian operator

```
// constructors
BASIS basis(WAV);
BOXS geometry(fnameBox);
VECSPACE_SPARSE vecspaces(basisp, geometry);
VECSPACE_DENSE  vecspaced(basisp, geometry.extent());

BIVEC_SPARSE VEC1(vecspaces, VEC_BRA);
BIVEC_SPARSE VEC2(vecspaces, VEC_BRA);
BIVEC_DENSE  vec1(vecspaced, VEC_BRA);
BIVEC_DENSE  vec2(vecspaced, VEC_BRA);

LAPLACIAN mat(vecspaces);

// input data
storePolyDenseTopLevel(VEC1, vec1, function);

// convert to sparse
convertToSparse(VEC1, vec1);

// VEC2 += mat * VEC1;
AddMult(VEC2, mat, VEC1);

// convert to dense
convertToDense(VEC2, vec1);

// plot
string fnameOut = "denseout.dat";
plotBox(fnameOut, vec1);
```

Generic Algorithm

CG

Supporting Objects

Functional	Constraint	Boundary	Convergence
• Information on the energy functional used • Easy implementation of new fucntionals • get_gradient() • get_dE_2nd_order_corr()	• Information on the constraints used • Lagrange matrix • apply() • modify_gradient()	• Information on the boundaries within unit cell • Information on the crystal periodicity • apply()	• Information on the convergence criteria • apply()

```
function CG_minimization {
  boundary.apply(s);
  constraint.apply(s);
  h = functional.gradient(s);
  h = constraint.modify_gradient(h, s); // get lambda
  boundary.apply(h);
  g = h;
  dE = functional.dE_2nd_order_corr(s, h, g, lambda);
  s = s + h * dE;
  constraint.apply(s);

  until(converged) {

    g = functional.gradient(s);
    g = constraint.modify_gradient(g, s); // get lambda
    boundary.apply(g);
    h = h * (<g|g>/<gold|gold>) - g;
    dE = functional.dE_2nd_order_corr(s, h, g, lambda);
    s = s + h * dE;
    constraint.apply(s);
  }
}

class boundary {
  apply(s) {
    // set s to zero outside domain
  }
};

class constraint {
  modify_gradient(hs, ss) {
    states_unpartitioned su(s);
    states_unpartitioned hu(h);
    // actual code exploits symetry
    // only one row needed
    lambda(j,i) = InnerProduct(su(j),hu(i));
    hu = hu - lambda(j,i) su(j)
  }
  apply(ss) {
    // apply symmetric orthogonalization to ss in place.
  }
};
```

```
class functional {
  gradient(hs, ss) {
    hs = wavelet_Hamiltonian_functor(ss);
  }
  dE_2nd_order_corr(ss, hs, gs, constraint, dE) {
    // H represents hamiltonian functor in get_gradient
    dE = <gs|gs> / (<hs | H | hs> - lambda(i,i) <hs|hs>);
  }
};

class states_unpartitioned {
  int nstates;
  TinyI ncells;
  int superindex(nstate, cell) { } // map indices
  int nstate(superindex) { }
  TinyI cell(superindex) { }
  // algebra on states incorporating shift between cells
  // inner product
  // overlap matrix
};
```