GPU Teaching Kit

Accelerated Computing

ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Module 15 - Application Case Study – Advanced MRI Reconstruction

Lecture 15.1 – Non-Cartesian MRI Reconstruction

Objective

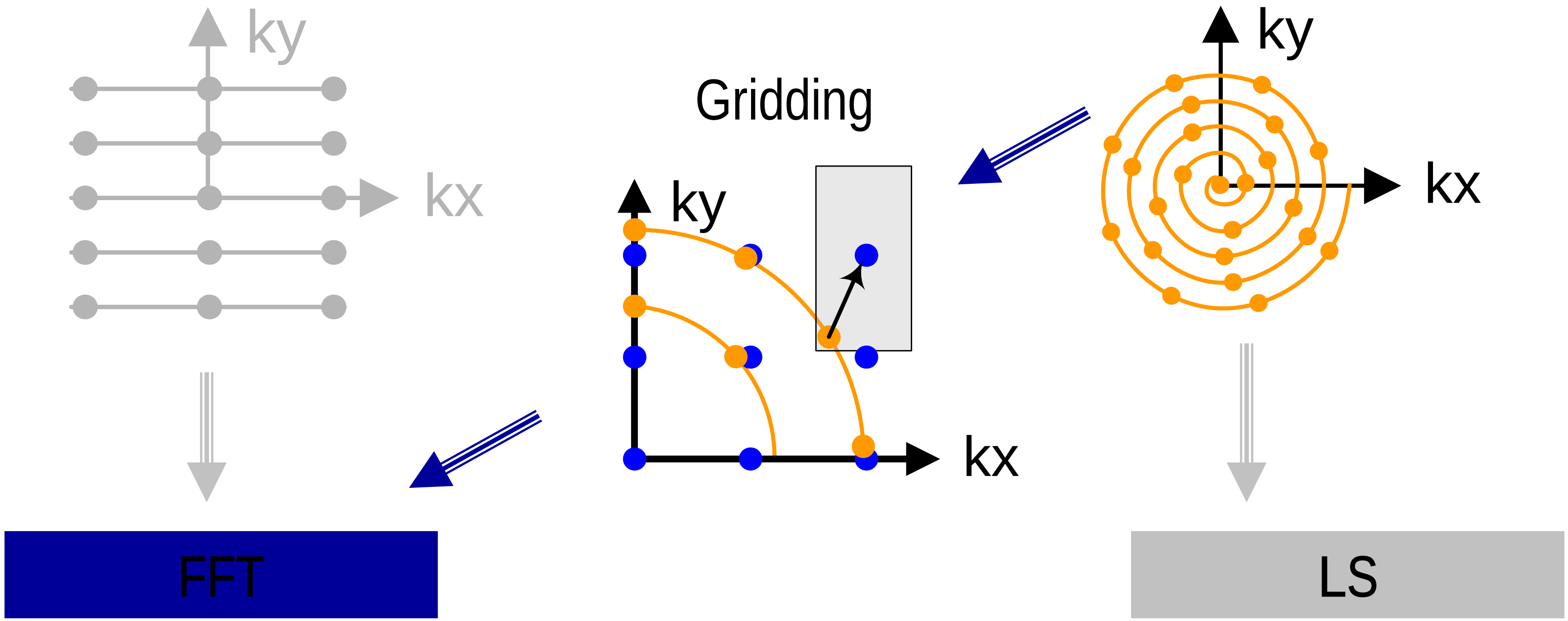
- To learn how to apply parallel programming techniques to an application
  - Determining parallelism structure
  - Loop transformations
  - Memory layout considerations
  - Validation

Cartesian vs. Non-Cartesian MRI Scan

$$\hat{m}(\mathbf{r}) = \sum_j W(\mathbf{k}_j) s(\mathbf{k}_j) e^{i2\pi \mathbf{k}_j \cdot \mathbf{r}}$$

Cartesian Scan Data

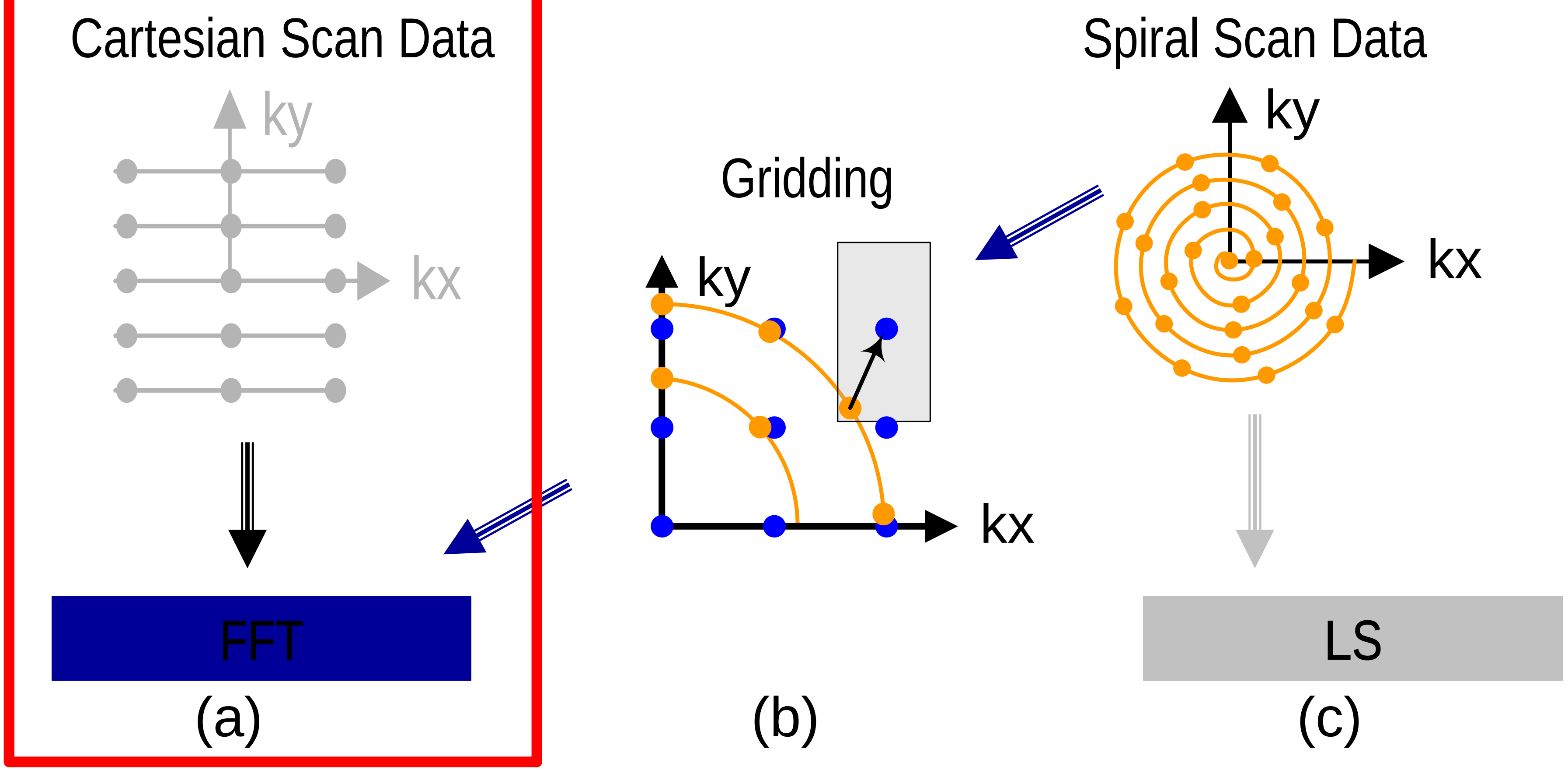
Spiral Scan Data



(a) (b) (c)

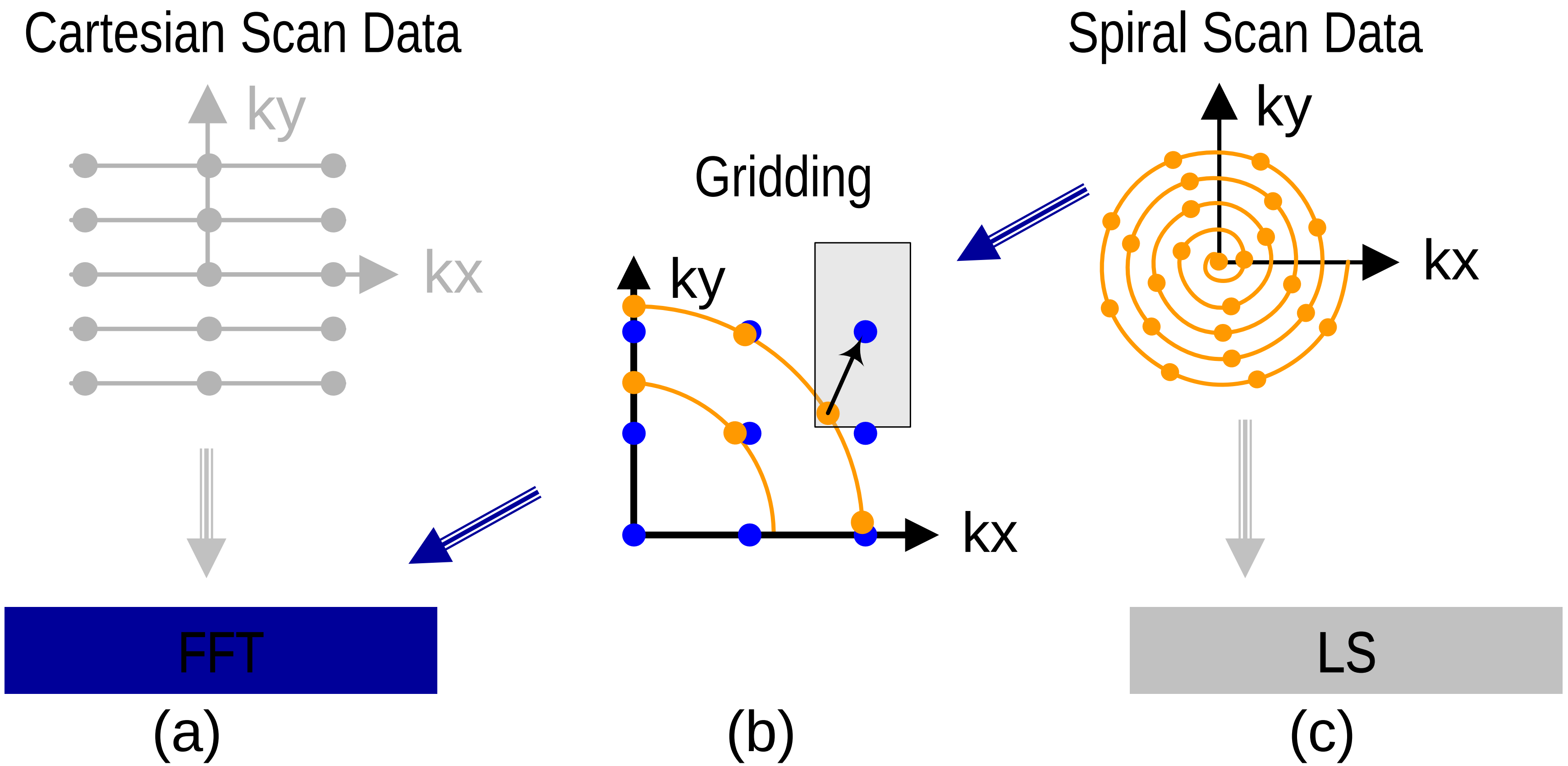
Cartesian vs. Non-Cartesian MRI Scan

$$\hat{m}(\mathbf{r}) = \sum_j W(\mathbf{k}_j) s(\mathbf{k}_j) e^{i2\pi \mathbf{k}_j \cdot \mathbf{r}}$$



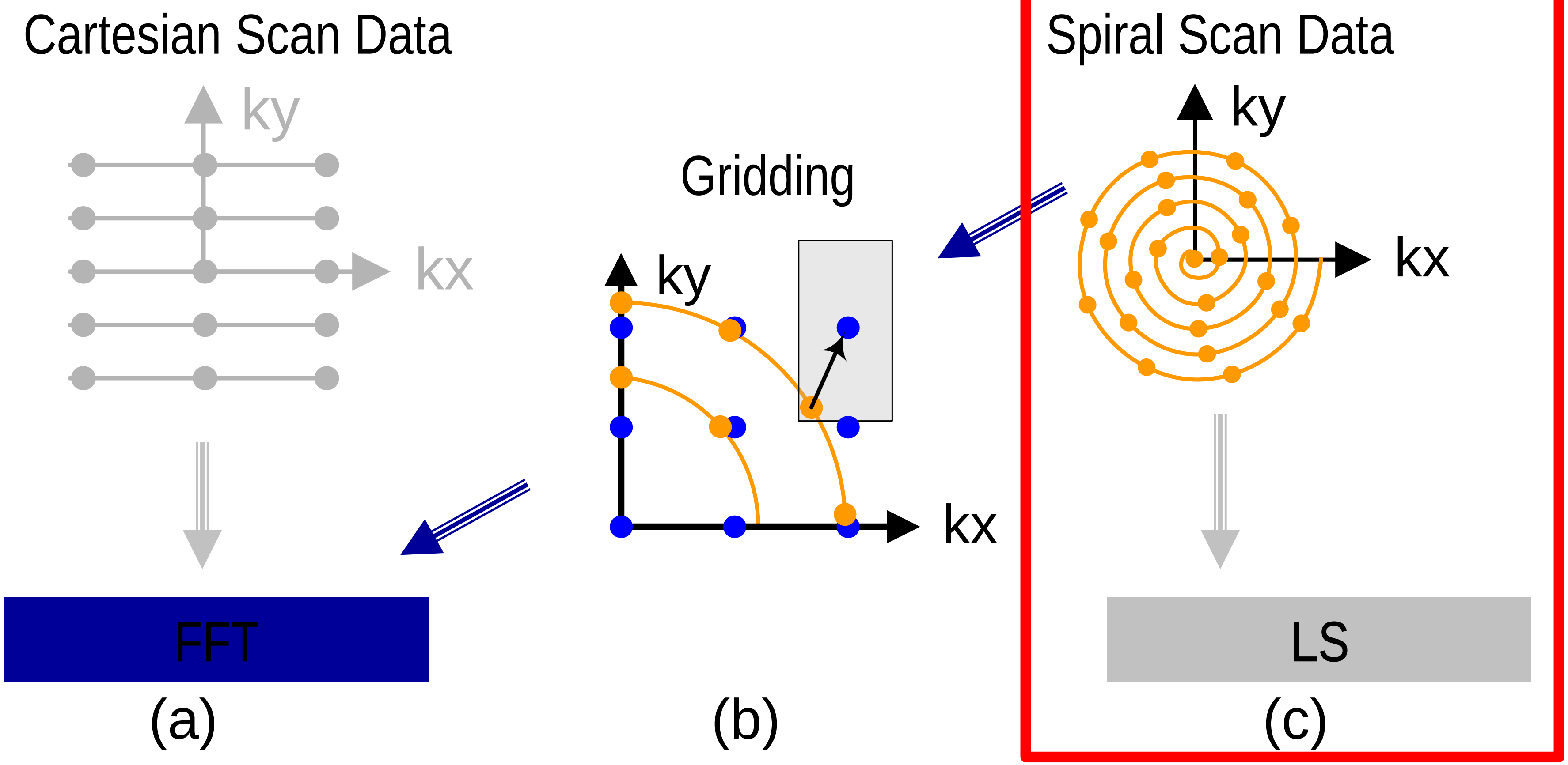
Cartesian vs. Non-Cartesian MRI Scan

$$\hat{m}(\mathbf{r}) = \sum_j W(\mathbf{k}_j) s(\mathbf{k}_j) e^{i2\pi \mathbf{k}_j \cdot \mathbf{r}}$$

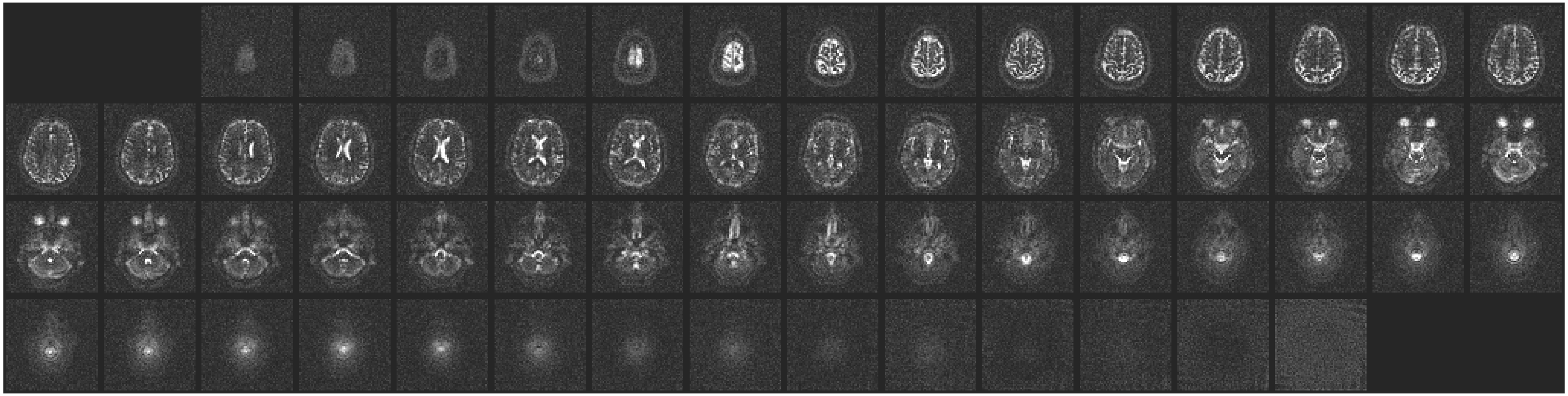


Cartesian vs. Non-Cartesian MRI Scan

$$\hat{m}(\mathbf{r}) = \sum_j W(\mathbf{k}_j) s(\mathbf{k}_j) e^{i2\pi \mathbf{k}_j \cdot \mathbf{r}}$$

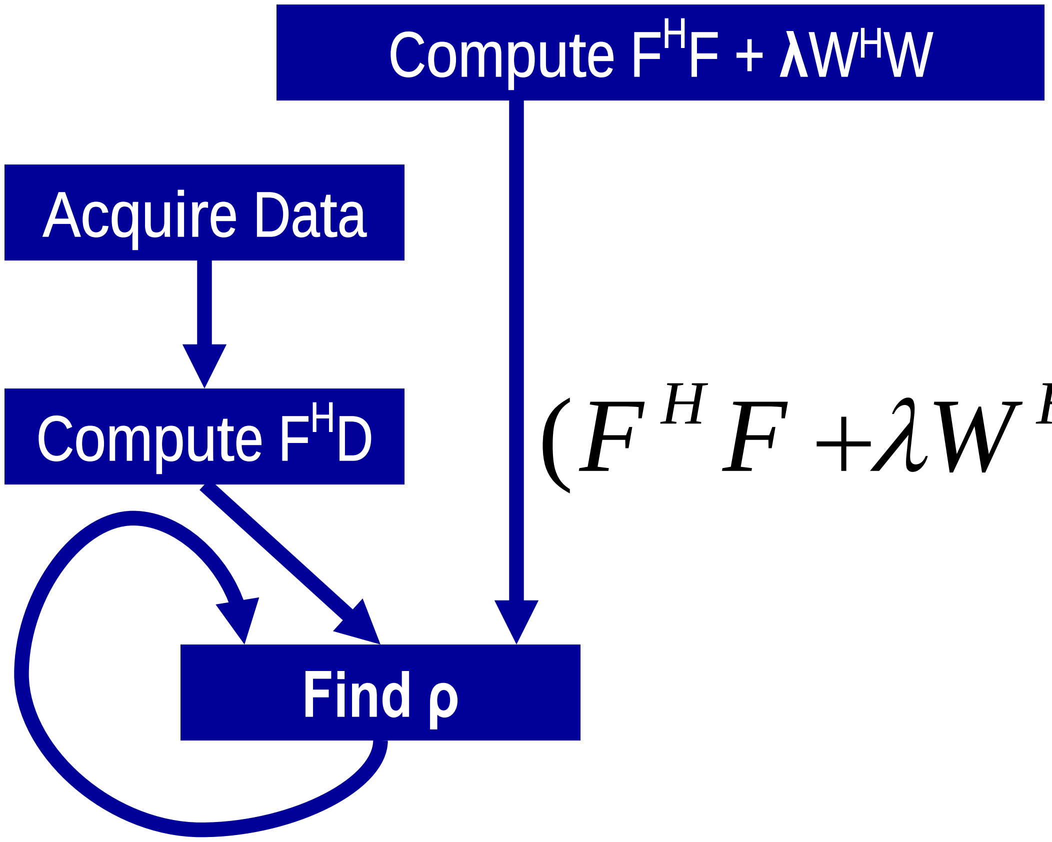


Non-Cartesian Scan



Courtesy of Keith Thulborn and Ian Atkinson, Center for MR Research, University of Illinois at Chicago

An Iterative Solver Based Approach to Image Reconstruction



$$(F^H F + \lambda W^H W)\rho = F^H D$$

Computation of Q and FHD

```
for (m = 0; m < M; m++) {

  phiMag[m] = rPhi[m]*rPhi[m] +
              iPhi[m]*iPhi[m];

  for (n = 0; n < N; n++) {
    expQ = 2*PI*(kx[m]*x[n] +
                 ky[m]*y[n] +
                 kz[m]*z[n]);

    rQ[n] +=phiMag[m]*cos(expQ);
    iQ[n] +=phiMag[m]*sin(expQ);
  }
}
```

(a) Q computation

```
for (m = 0; m < M; m++) {

  rMu[m] = rPhi[m]*rD[m] +
           iPhi[m]*iD[m];
  iMu[m] = rPhi[m]*iD[m] –
           iPhi[m]*rD[m];

  for (n = 0; n < N; n++) {
    expFhD = 2*PI*(kx[m]*x[n] +
                   ky[m]*y[n] +
                   kz[m]*z[n]);

    cArg = cos(expFhD);
    sArg = sin(expFhD);

    rFhD[n] +=  rMu[m]*cArg –
                iMu[m]*sArg;
    iFhD[n] +=  iMu[m]*cArg +
                rMu[m]*sArg;
  }
}
```

(b) $F^H D$ computation

GPU Teaching Kit

Accelerated Computing