



# Fast-Fourier Optimization

High-Contrast Imaging and the Search for Exoplanets

Robert J. Vanderbei

May 8, 2013

Optimization Days  
GERAD  
HEC Montréal

# The Plan...

Brief Review of the *Fast Fourier Transform* (FFT)

FFT as a Sparse Factorization of a Dense Matrix

Embedding FFT as an Oracle in Derivative-Free Optimizer

Encoding FFT as Sparse Constraints in a Derivative-Based Optimizer

Some Applications:

- Pupil-Mask Coronagraphy
- Compressed Sensing
- Sparse Detection

# The Message...

FFT  $\subset$  Sparse Factorization:  $A = A_1 A_2 \cdots A_k$

# Fourier Transform

Fourier Transform:

$$\widehat{f}(\xi) = \int_{-\infty}^{\infty} e^{2\pi i x \xi} f(x) dx, \quad \xi \in \mathbb{R}.$$

Optimizer's *curse of dimensionality*: An optimization problem whose variables are a function  $f(x)$ ,  $x \in \mathbb{R}$ , and whose constraints are given on the Fourier transform is an  $\infty \times \infty$ -dimensional problem.

Some “tricks” are used to address this curse.

A first step is to assume (as is often physically realistic) that the function  $f$  has compact support, say, on  $[0, 1]$ :

$$\widehat{f}(\xi) = \int_0^1 e^{2\pi i x \xi} f(x) dx, \quad \xi \in \mathbb{R}.$$

The Nyquist-Shannon Sampling Theorem then says that the Fourier transform is completely characterized by its values at  $\xi = j\Delta\xi$ ,  $j = 0, 1, 2, \dots$ , and  $\Delta\xi = 1$ .

A second step is to discretize the integral:

$$\widehat{f}_j = \sum_{k=0}^{n-1} e^{2\pi i k \Delta x j \Delta \xi} f_k \Delta x, \quad 0 \leq j < m.$$

Complexity:  $O(mn)$

# Fast-Fourier Transform (FFT)

Recall one-dimensional Fourier transform:

$$\widehat{f}(\xi) = \int_0^1 e^{2\pi i x \xi} f(x) dx.$$

and its discrete approximation:

$$\widehat{f}_j = \sum_{k=0}^{n-1} e^{2\pi i k \Delta x j \Delta \xi} f_k \Delta x, \quad 0 \leq j < m.$$

Suppose that  $n$  and  $m$  can be factored:

$$n = n_0 n_1 \quad \text{and} \quad m = m_0 m_1.$$

If we now decompose our sequencing indices  $k$  and  $j$  into

$$k = n_0 k_1 + k_0 \quad \text{and} \quad j = m_0 j_1 + j_0,$$

we get

$$\widehat{f}_{j_0, j_1} = \sum_{k_0=0}^{n_0-1} \sum_{k_1=0}^{n_1-1} e^{2\pi i n_0 k_1 \Delta x m_0 j_1 \Delta \xi} e^{2\pi i n_0 k_1 \Delta x j_0 \Delta \xi} e^{2\pi i k_0 \Delta x (m_0 j_1 + j_0) \Delta \xi} f_{k_0, k_1} \Delta x.$$

# Fast-Fourier Transform (FFT)

$$\widehat{f}_{j_0, j_1} = \sum_{k_0=0}^{n_0-1} \sum_{k_1=0}^{n_1-1} e^{2\pi i n_0 k_1 \Delta x m_0 j_1 \Delta \xi} e^{2\pi i n_0 k_1 \Delta x j_0 \Delta \xi} e^{2\pi i k_0 \Delta x (m_0 j_1 + j_0) \Delta \xi} f_{k_0, k_1} \Delta x.$$

We want the first exponential factor to evaluate to one. To make that happen, we assume that  $n_0 m_0 \Delta x \Delta \xi$  is an integer. With the first exponential factor gone, we can write down a *two-step* algorithm

$$g_{j_0, k_0} = \sum_{k_1=0}^{n_1-1} e^{2\pi i n_0 k_1 \Delta x j_0 \Delta \xi} f_{k_0, k_1} \Delta x, \quad 0 \leq j_0 < m_0, \quad 0 \leq k_0 < n_0,$$

$$\widehat{f}_{j_0, j_1} = \sum_{k_0=0}^{n_0-1} e^{2\pi i k_0 \Delta x (m_0 j_1 + j_0) \Delta \xi} g_{j_0, k_0}, \quad 0 \leq j_0 \leq m_0, \quad 0 \leq j_1 \leq m_1.$$

# Complexity

The number of multiply/adds required for this two-step algorithm is

$$n_0 n_1 m_0 + m_0 m_1 n_0 = mn \left( \frac{1}{m_1} + \frac{1}{n_1} \right).$$

If  $m \approx n$  and  $m_1 \approx n_1 \approx \sqrt{n}$ , the complexity simplifies to

$$2n\sqrt{n}.$$

Compared to the one-step algorithm, which takes  $n^2$  multiply/adds, this two-step algorithm gives an improvement of a factor of  $\sqrt{n}/2$ . Also, if  $m$  is much smaller than  $n$ , we get further improvement over the full  $n \times n$  case.

Of course, if  $m_0$ ,  $m_1$ ,  $n_0$ , and  $n_1$  can be further factored, then this two-step algorithm can be extended recursively.

For the *FFT*,  $m$  and  $n$  are chosen to be a power of 2. In this case, the recursively applied algorithm is an  $n \log_2 n$  algorithm.

# Fourier Optimization

$$\begin{aligned} \text{optimize} \quad & \sum_{k=0}^{n-1} c_k f_k \\ \text{subject to} \quad & \widehat{f}_j = \sum_{k=0}^{n-1} e^{2\pi i k \Delta x j \Delta \xi} f_k \Delta x, \quad j \in \mathcal{J} \\ & |\widehat{f}_j| \leq \epsilon, \quad j \in \mathcal{J}. \end{aligned}$$

The set  $\mathcal{J}$  is often “small”.

The “magnitude” of a complex number  $x + iy$  is  $\sqrt{x^2 + y^2}$ .

Hence, the problem, as formulated, is a *second-order cone programming* (SOCP) problem.

Often symmetry implies that  $\widehat{f}$  is real (i.e., the imaginary part vanishes).

In this case, it is easy to convert the SOCP to a *linear programming* (LP) problem.

The Jacobian of the linear operator defining the discretized Fourier transform is a *dense*  $m \times n$ -matrix.

# Fast Fourier Optimization (Oracle Version)

$$\begin{aligned} &\text{optimize} && \sum_{k=0}^{n-1} c_k f_k \\ &\text{subject to} && \hat{f} = \text{fftw}(f) \\ &&& |\hat{f}_j| \leq \epsilon, \quad j \in \mathcal{J}. \end{aligned}$$

*fftw* stands for *Fastest Fourier Transform in the West*.

It is regarded as the fastest (and most general) of the fft algorithms available.

Problem is still a linear programming problem (or SOCP depending).

But, the routine *fftw* (and other similar oracles) do not provide the Jacobian matrix of first derivatives.

Hence, this formulation can only be solved using *derivative-free* optimization methods.

Furthermore, it may be serious overkill to compute  $\hat{f}_j$  for *all*  $j = 1, 2, \dots, n$ .

# Fast Fourier Optimization (A Better Way)

$$\text{optimize } \sum_{k=0}^{n-1} c_k f_k$$

$$\text{subject to } g_{j_0, k_0} = \sum_{k_1=0}^{n_1-1} e^{2\pi i n_0 k_1 \Delta x j_0 \Delta \xi} f_{k_0, k_1} \Delta x, \quad 0 \leq j_0 < m_0, \quad 0 \leq k_0 < n_0,$$

$$\widehat{f}_{j_0, j_1} = \sum_{k_0=0}^{n_0-1} e^{2\pi i k_0 \Delta x (m_0 j_1 + j_0) \Delta \xi} g_{j_0, k_0}, \quad 0 \leq j_0 \leq m_0, \quad 0 \leq j_1 \leq m_1,$$

$$|\widehat{f}_{j_0, j_1}| \leq \epsilon, \quad j = m_0 j_1 + j_0 \in \mathcal{J}.$$

The constraint matrix  $A$  is *sparse* (details later).

As with FFT, this “factorization” can be continued.

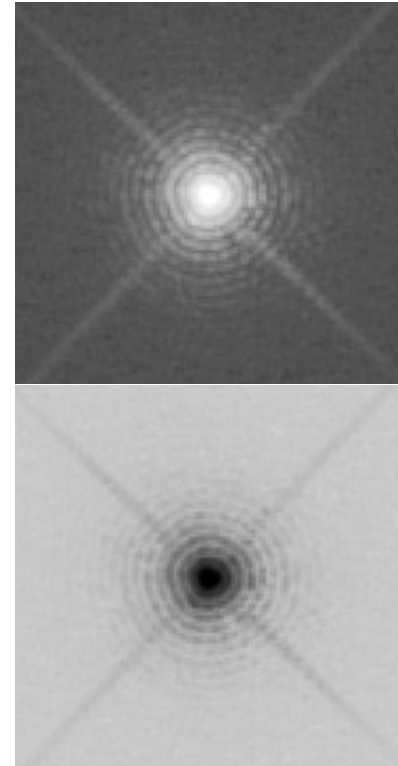
# Applications

# High-Contrast Imaging for Planet-Finding

Build a telescope capable of finding Earth-like planets around nearby Sun-like stars.

Problem is hard:

- Star is  $10^{10}$  times brighter than the planet.
- Angular separation is small  $\approx 0.05$  arcseconds.
- Light is a wave: the star is not a pinpoint of light—it has a *diffraction pattern*.
- Light is photons  $\Rightarrow$  Poisson statistics.



The diffraction pattern is the magnitude-squared of the *Fourier transform* of the telescope's *pupil*.

# Pupil Masking/Apodization



# Pupil Masking/Apodization



# Pupil Apodization

Let  $f(x, y)$  denote the transmissivity (i.e., *apodization*) at location  $(x, y)$  on the surface of a filter placed over the pupil of a telescope.

The *electromagnetic field* in the image plane of such a telescope associated with an on-axis point source (i.e., a star) is proportional to the 2D Fourier transform of the apodization  $f$ :

$$\hat{f}(\xi, \eta) = \iint e^{2\pi i(x\xi + y\eta)} f(x, y) dx dy.$$

The *intensity* of the light in the image is proportional to the *magnitude squared* of  $\hat{f}$ .

Assuming that the underlying telescope has a circular opening of radius one, we impose the following constraint on  $f$ :

$$f(x, y) = 0 \quad \text{for} \quad x^2 + y^2 > 1.$$

# Optimized Apodizations

Maximize: *light throughput*

Subject to: constraint that almost no light reaches a given *dark zone*  $\mathcal{D}$  and

other *structural* constraints:

$$\begin{aligned} \text{maximize} \quad & \int_{\square} \int f(x, y) dx dy \quad \left( = \widehat{f}(0, 0) \right) \\ \text{subject to} \quad & \left| \widehat{f}(\xi, \eta) \right| \leq \varepsilon \widehat{f}(0, 0), \quad (\xi, \eta) \in \mathcal{D}, \\ & f(x, y) = 0, \quad x^2 + y^2 > 1, \\ & 0 \leq f(x, y) \leq 1, \quad \text{for all } x, y. \end{aligned}$$

Here,  $\varepsilon$  is a small positive constant (on the order of  $10^{-5}$ ).

In general, the Fourier transform  $\widehat{f}$  is complex valued.

This optimization problem has a *linear objective* function and both *linear* and *second-order cone* constraints.

Hence, a discretized version can be solved (to a *global optimum*).

# Exploiting Symmetry

Assuming that the filter can be symmetric with respect to reflection about both axes (note: sometimes not possible), the Fourier transform can be written as

$$\hat{f}(\xi, \eta) = 4 \int_0^1 \int_0^1 \cos(2\pi x\xi) \cos(2\pi y\eta) f(x, y) dx dy.$$

In this case, the Fourier transform is real and so the second-order cone constraints can be replaced with a pair of inequalities,

$$-\varepsilon \hat{f}(0, 0) \leq \hat{f}(\xi, \eta) \leq \varepsilon \hat{f}(0, 0),$$

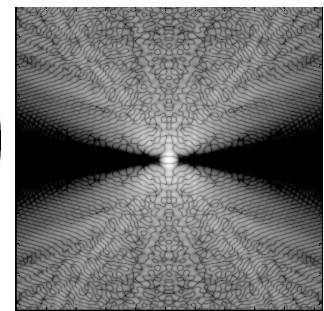
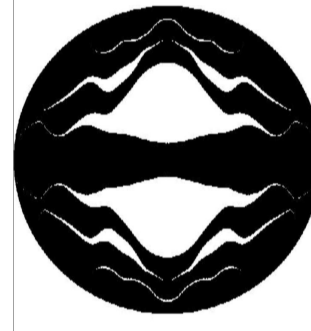
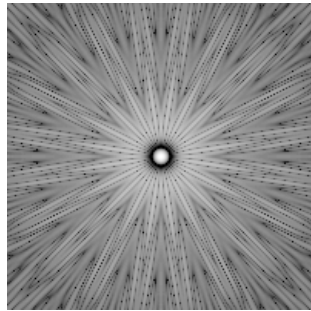
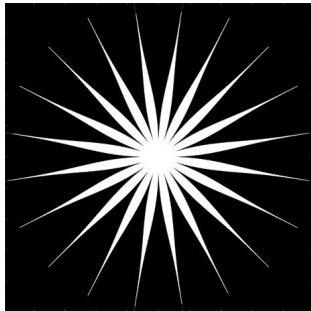
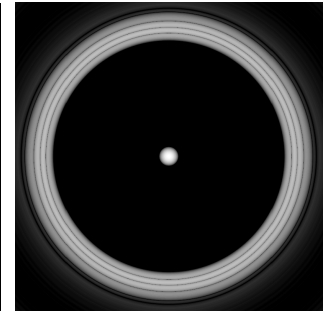
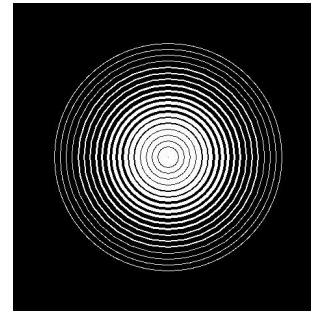
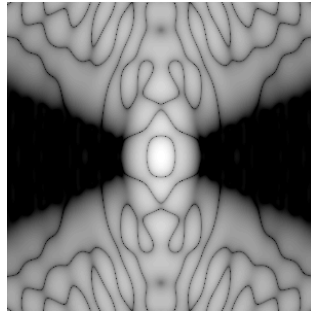
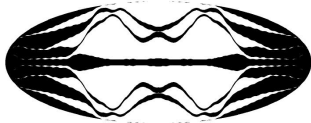
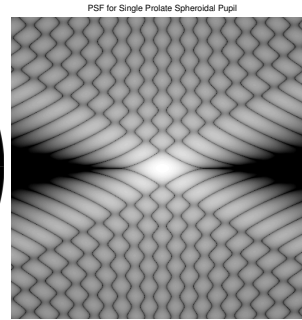
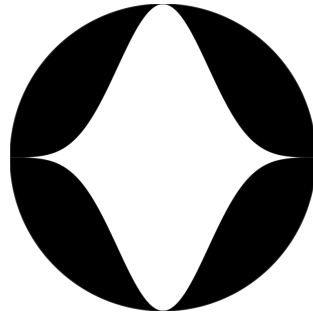
making the problem an *infinite dimensional linear programming problem*.

(Probabilist's) *Curse of Dimensionality*:  $2 > 1$ .

# Russell Crowe Comments on the Curse



# Potpourri of Pupil Masks



# Discretization

Consider a two-dimensional Fourier transform

$$\widehat{f}(\xi, \eta) = 4 \int_0^1 \int_0^1 \cos(2\pi x\xi) \cos(2\pi y\eta) f(x, y) dx dy.$$

Its discrete approximation can be computed as

$$\widehat{f}_{j_1, j_2} = 4 \sum_{k_2=1}^n \sum_{k_1=1}^n \cos(2\pi x_{k_1} \xi_{j_1}) \cos(2\pi y_{k_2} \eta_{j_2}) f_{k_1, k_2} \Delta x \Delta y, \quad 1 \leq j_1, j_2 \leq m,$$

where

$$\begin{aligned} x_k &= (k - 1/2)\Delta x, & 1 \leq k \leq n, \\ y_k &= (k - 1/2)\Delta y, & 1 \leq k \leq n, \\ \xi_j &= (j - 1/2)\Delta \xi, & 1 \leq j \leq m, \\ \eta_j &= (j - 1/2)\Delta \eta, & 1 \leq j \leq m, \\ f_{k_1, k_2} &= f(x_{k_1}, y_{k_2}), & 1 \leq k_1, k_2 \leq n, \\ \widehat{f}_{j_1, j_2} &\approx \widehat{f}(\xi_{j_1}, \eta_{j_2}), & 1 \leq j_1, j_2 \leq m. \end{aligned}$$

*Complexity:  $m^2 n^2$ .*

# A Clever (and Trivial!) Idea

The obvious brute force calculation requires  $m^2 n^2$  operations.

However, we can “factor” the double sum into a nested pair of sums.

Introducing new variables that represent the inner sum, we get:

$$g_{j_1, k_2} = 2 \sum_{k_1=1}^n \cos(2\pi x_{k_1} \xi_{j_1}) f_{k_1, k_2} \Delta x, \quad 1 \leq j_1 \leq m, \quad 1 \leq k_2 \leq n,$$

$$\widehat{f}_{j_1, j_2} = 2 \sum_{k_2=1}^n \cos(2\pi y_{k_2} \eta_{j_2}) g_{j_1, k_2} \Delta y, \quad 1 \leq j_1, j_2 \leq m,$$

Formulated this way, the calculation requires only  $mn^2 + m^2 n$  operations.

# Brute Force vs Clever Approach

On the following page we show two AMPL model formulations of this problem.

On the left is the version expressed in the straightforward one-step manner.

On the right is the AMPL model for the same problem but with the Fourier transform expressed as a pair of transforms—the so-called *two-step process*.

The dark zone  $\mathcal{D}$  is a pair of sectors of an annulus with inner radius 4 and outer radius 20.

Except for the resolution, the two models produce the same result.

# Two AMPL Models

```
param rho0 := 4;      param rho1 := 20;
param m := 35;      # discretization parameter
param n := 150;     # discretization parameter
param dx := 1/(2*n);  param dy := dx;

set Xs := setof {j in 0.5..n-0.5 by 1} j/(2*n);
set Ys := Xs;
set Pupil :=
    setof {x in Xs, y in Ys: x^2+y^2<0.25} (x,y);
set Xis := setof {j in 0..m} j*rho1/m;
set Etas := Xis;
set DarkHole := setof {xi in Xis, eta in Etas:
    xi^2+eta^2>=rho0^2 &&
    xi^2+eta^2<=rho1^2 &&
    eta <= xi } (xi,eta);

var f {(x,y) in Pupil} >= 0, <= 1;

var fhat {xi in Xis, eta in Etas};

maximize area: sum {(x,y) in Pupil} f[x,y]*dx*dy;

subject to fhat_def {xi in Xis, eta in Etas}:
    fhat[xi,eta] = 4*sum {(x,y) in Pupil}
        f[x,y]*cos(2*pi*x*xi)
            *cos(2*pi*y*eta)*dx*dy;
subject to sidelobe_pos {(xi,eta) in DarkHole}:
    fhat[xi,eta] <= 10^(-5)*fhat[0,0];
subject to sidelobe_neg {(xi,eta) in DarkHole}:
    -10^(-5)*fhat[0,0] <= fhat[xi,eta];

solve;
```

```
param rho0 := 4;      param rho1 := 20;
param m := 35;      # discretization parameter
param n := 1000;     # discretization parameter
param dx := 1/(2*n);  param dy := dx;

set Xs := setof {j in 0.5..n-0.5 by 1} j/(2*n);
set Ys := Xs;
set Pupil :=
    setof {x in Xs, y in Ys: x^2+y^2 < 0.25} (x,y);
set Xis := setof {j in 0..m} j*rho1/m;
set Etas := Xis;
set DarkHole := setof {xi in Xis, eta in Etas:
    xi^2+eta^2>=rho0^2 &&
    xi^2+eta^2<=rho1^2 &&
    eta <= xi } (xi,eta);

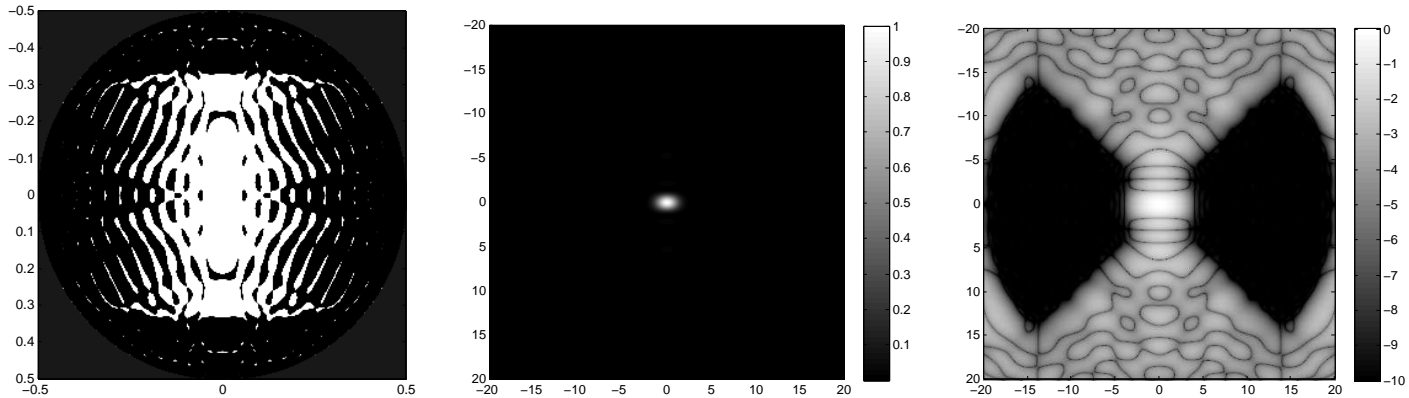
var f {(x,y) in Pupil} >= 0, <= 1;
var g {xi in Xis, y in Ys};
var fhat {xi in Xis, eta in Etas};

maximize area: sum {(x,y) in Pupil} f[x,y]*dx*dy;

subject to g_def {xi in Xis, y in Ys}:
    g[xi,y] = 2*sum {x in Xs: (x,y) in Pupil}
        f[x,y]*cos(2*pi*x*xi)*dx;
subject to fhat_def {xi in Xis, eta in Etas}:
    fhat[xi,eta] = 2*sum {y in Ys}
        g[xi,y]*cos(2*pi*y*eta)*dy;
subject to sidelobe_pos {(xi,eta) in DarkHole}:
    fhat[xi,eta] <= 10^(-5)*fhat[0,0];
subject to sidelobe_neg {(xi,eta) in DarkHole}:
    -10^(-5)*fhat[0,0] <= fhat[xi,eta];

solve;
```

# Optimal Solution



*Left.* The optimal apodization found by either of the models shown on previous slide.

*Center.* Plot of the star's image (using a linear stretch).

*Right.* Logarithmic plot of the star's image (black =  $10^{-10}$ ).

Notes:

- The “apodization” turns out to be purely opaque and transparent (i.e., a mask).
- The mask has “islands” and therefore must be laid on glass.

# Close Up

Brute force with  $n = 150$



Two-step with  $n = 1000$



# Summary Problem Stats

Comparison between a few sizes of the one-step and two-step models.

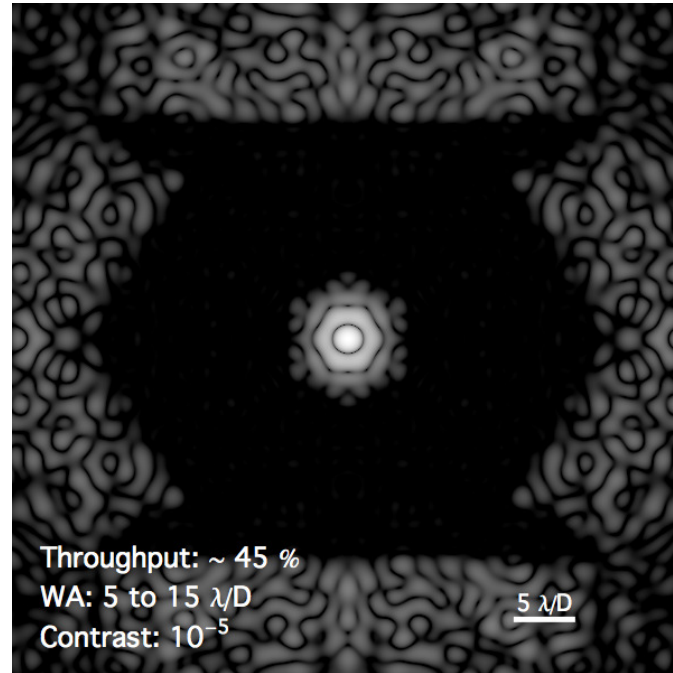
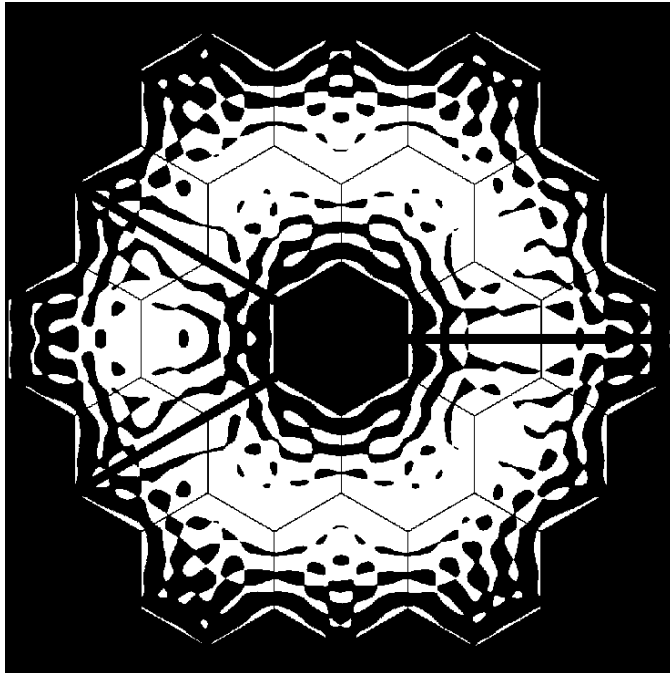
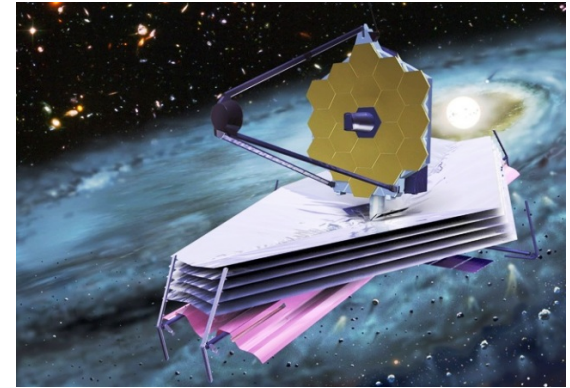
Problem-specific stats.

Model	$n$	$m$	constraints	variables	nonzeros	arith. ops.
One step	150	35	976	17,672	17,247,872	17,196,541,336
One step	250	35	*	*	*	*
Two step	150	35	7,672	24,368	839,240	3,972,909,664
Two step	500	35	20,272	215,660	7,738,352	11,854,305,444
Two step	1000	35	38,272	822,715	29,610,332	23,532,807,719

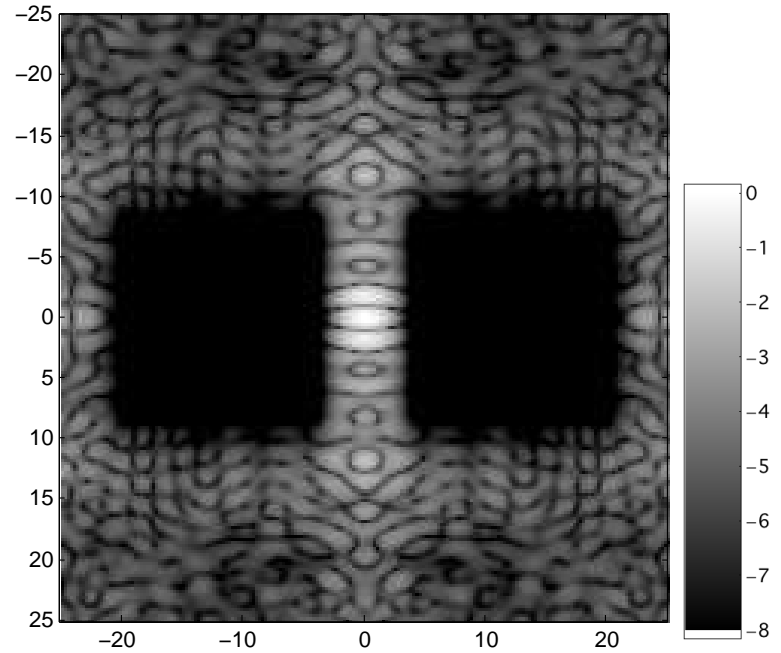
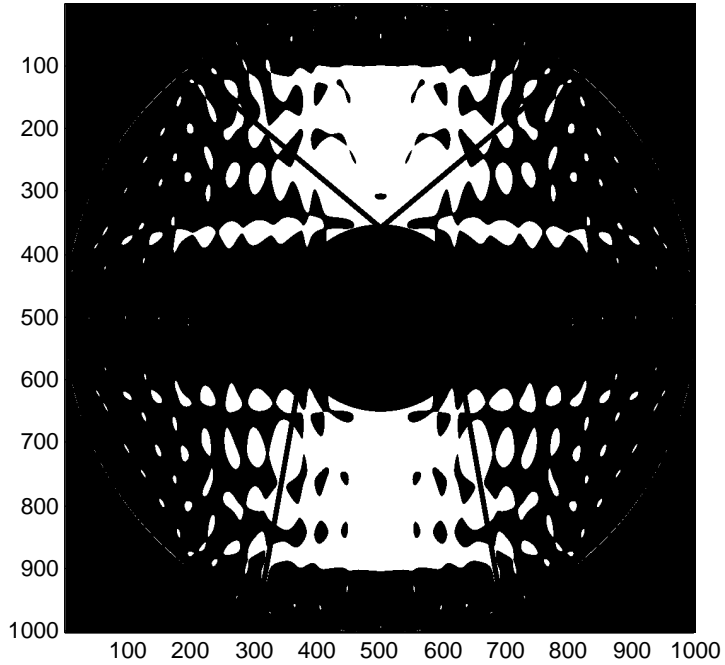
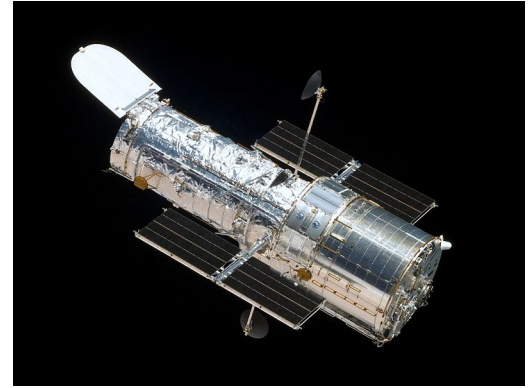
Hardware/Solution-specific performance comparison data.

Model	$n$	$m$	iterations	primal objective	dual objective	cpu time (sec)
One step	150	35	54	0.05374227247	0.05374228041	1380
One step	250	35	*	*	*	*
Two step	150	35	185	0.05374233071	0.05374236091	1064
Two step	500	35	187	0.05395622255	0.05395623990	4922
Two step	1000	35	444	0.05394366337	0.05394369256	26060

# James Webb Space Telescope (JWST)



# Repurposed NRO Spy Satellite



# Matrix Notation

Let

$$F := [f_{k_1, k_2}], \quad G := [g_{j_1, k_2}], \quad \widehat{F} := [\widehat{f}_{j_1, j_2}], \quad \text{and} \quad K := [\kappa_{j_1, k_1}],$$

where  $K$  denotes the  $m \times n$  *Fourier kernel* matrix whose elements are

$$\kappa_{j_1, k_1} = 2 \cos(2\pi x_{k_1} \xi_{j_1}) \Delta x.$$

The two-dimensional Fourier transform  $\widehat{F}$  can be written simply as

$$\widehat{F} = KFK^T$$

and the computation of the transform in two steps is just the statement that the two matrix multiplications can (*and should!*) be done separately:

$$G = KF$$

$$\widehat{F} = GK^T.$$

# MATLAB

Brute force method:

```
for j1 in 1:m
    for j2 in 1:m
        fhat(j1,j2) = 0;
        for k1 in 1:n
            for k2 in 1:n
                fhat(j1,j2) = fhat(j1,j2) + ...
                    4 * cos(2*pi*x(k1)*xi(j1)) * ...
                    cos(2*pi*y(k2)*eta(j2)) * ...
                    f(k1,k2)*dx*dy;
            end
        end
    end
end
```

Two-step method:

```
K = 2 * cos(2*pi*xi*x')*dx;
fhat = K*f*K';
```

# Linear Programming

Clever Idea = Matrix Sparsification

Linear programming algorithms solve problems in this form:

$$\begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax = b, \\ & x \geq 0, \end{array}$$

where  $b$  and  $c$  are given vectors and  $A$  is a given matrix.

Of course,  $x$  is a vector.

AMPL converts a problem from its “natural” formulation to this paradigm and then hands it off to a solver.

**IMPORTANT:** The Fourier transform is a *linear operator*. Some suggest invoking the *fast Fourier transform (FFT)* to get maximum efficiency automatically. No can do. We need  $A$ , which is the matrix representing the Fourier transform (fast or otherwise). In other words, the optimization algorithm needs the *Jacobian* of the linear operator.

# The Jacobian

Let  $f_j$ ,  $g_j$ , and  $\hat{f}_j$  denote the column vectors of matrices  $F$ ,  $G$ , and  $\hat{F}$ :

$$F = [f_1 \cdots f_n], \quad G = [g_1 \cdots g_n], \quad \hat{F} = [\hat{f}_1 \cdots \hat{f}_m].$$

We can list the elements of  $F$ ,  $G$  and  $\hat{F}$  in column vectors:

$$\text{vec}(F) = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \quad \text{vec}(G) = \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}, \quad \text{vec}(\hat{F}) = \begin{bmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_m \end{bmatrix}.$$

It is straightforward to check that

$$\text{vec}(G) = \begin{bmatrix} K & & \\ & \cdots & \\ & & K \end{bmatrix} \text{vec}(F)$$

and that

$$\text{vec}(\hat{F}) = \begin{bmatrix} \kappa_{1,1}I & \cdots & \kappa_{1,n}I \\ \vdots & & \vdots \\ \kappa_{m,1}I & \cdots & \kappa_{m,n}I \end{bmatrix} \text{vec}(G).$$

# One-Step Method

$$Ax = b$$



$$\left[ \begin{array}{ccc|c} \kappa_{1,1}K & \cdots & \kappa_{1,n}K & -I \\ \vdots & & \vdots & \cdots \\ \kappa_{m,1}K & \cdots & \kappa_{m,n}K & -I \\ \hline & \vdots & & \vdots \end{array} \right] \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ \hline \widehat{f}_1 \\ \vdots \\ \widehat{f}_m \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \vdots \end{bmatrix}$$

The big left block is a *dense*  $m^2 \times n^2$  matrix.



# Another Application: Compressive Sensing

**Hidden:** A large (length  $n$ ) but very sparse vector  $x^0$ .

**Observed:** A much shorter (length  $m$ ) vector  $y = Ax^0$ , where the matrix  $A$  can be specified as we like.

Recover  $x^0$  by solving optimization problem:

$$\begin{array}{ll} \text{minimize} & \|x\|_1 \\ \text{subject to} & Ax = y. \end{array}$$

Problem is converted to a linear programming problem in the usual manner:

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T(x^+ + x^-) \\ \text{subject to} & A(x^+ - x^-) = y \\ & x^+, x^- \geq 0. \end{array}$$

It is much more efficient to pack  $x^0$  and  $y$  into matrices  $X^0$  and  $Y$  and solve a related problem:

$$\begin{aligned} & \text{minimize} && \mathbf{1}^T(X^+ + X^-)\mathbf{1}^T \\ & \text{subject to} && A(X^+ - X^-)B^T = Y \\ & && X^+, X^- \geq 0. \end{aligned}$$

Here,  $A$  and  $B$  are specified as we like.

Assume that the total number of elements of  $X$  is  $n$  and of  $Y$  is  $m$ , where  $n$  and  $m$  are as before.

Computational experiments show that for  $n = 141 \times 142$  and  $m = 33 \times 34$ , the sparsified version of this problem solves about 100 times faster than the original.

# And Another Application: Sparse Detection

$$\begin{aligned} & \text{minimize} && \lambda \|X\|_1 + \|E\|_2^2 \\ & \text{subject to} && KXK^T + E = B, \\ & && X \geq 0. \end{aligned}$$

Here...

- $B$  is the observed “image”, which is the 2D Fourier Transform of the “true” image  $X$ .
- $E$  is a matrix of iid noise parameters (so-called *shot* noise).
- $K$  is an  $m \times n$  Fourier kernel.
- $\lambda$  is a positive parameter controlling the trade-off between the two terms of the objective function.

A sparse formulation:

$$\begin{aligned} & \text{minimize} && \lambda \|X\|_1 + \|E\|_2^2 \\ & \text{subject to} && XK^T - Y = 0, \\ & && KY + E = B, \\ & && X \geq 0. \end{aligned}$$

# And Another Application: Sparse Detection

$$\begin{aligned} & \text{minimize} && \lambda \|X\|_1 + \|E\|_2^2 \\ & \text{subject to} && KXK^T + E = B, \\ & && X \geq 0. \end{aligned}$$

Here...

- $B$  is the observed “image”, which is the 2D Fourier Transform of the “true” image  $X$ .
- $E$  is a matrix of iid noise parameters (so-called *shot* noise).
- $K$  is an  $m \times n$  Fourier kernel.
- $\lambda$  is a positive parameter controlling the trade-off between the two terms of the objective function.

A sparser formulation (based on  $K = K_1 K_2$ ):

$$\begin{aligned} & \text{minimize} && \lambda \|X\|_1 + \|E\|_2^2 \\ & \text{subject to} && XK_2^T - Y_2 && = 0, \\ & && Y_2 K_1^T - Y_1 && = 0, \\ & && K_2 Y_1 - Z_2 && = 0, \\ & && K_1 Z_2 + E = B, \\ & && X \geq 0. \end{aligned}$$

Sparse formulation:

$$\begin{aligned}
 & \text{minimize} && \lambda \|X\|_1 + \|E\|_2^2 \\
 & \text{subject to} && XK^T - Y = 0, \\
 & && KY + E = B, \\
 & && X \geq 0.
 \end{aligned}$$

Sparser formulation (based on  $K = K_1 K_2$ ):

$$\begin{aligned}
 & \text{minimize} && \lambda \|X\|_1 + \|E\|_2^2 \\
 & \text{subject to} && XK_2^T - Y_2 = 0, \\
 & && Y_2 K_1^T - Y_1 = 0, \\
 & && K_2 Y_1 - Z_2 = 0, \\
 & && K_1 Z_2 + E = B, \\
 & && X \geq 0.
 \end{aligned}$$

# References

- [1] R.J. Vanderbei, Han Liu, and Lie Wang. Optimization for Compressed Sensing: New Insights and Alternatives. *Submitted to MPC*, 2013.
- [2] R. Soummer, L. Pueyo, A. Sivaramakrishnan, and R.J. Vanderbei. Fast computation of lyot-style coronagraph propagation. *Optics Express*, 15(24):15935–15951, 2007.
- [3] A. Carlotti, R. J. Vanderbei, and N. J. Kasdin. Optimal pupil apodizations for arbitrary apertures. *Optics Express*, 19(27):26796–26809, 2011.
- [4] R. J. Vanderbei. Fast fourier optimization. *Math. Prog. Comp.*, 4(1):1–17, 2012.
- [5] N.J. Kasdin, R.J. Vanderbei, D.N. Spergel, and M.G. Littman. Extrasolar Planet Finding via Optimal Apodized and Shaped Pupil Coronagraphs. *Astrophysical Journal*, 582:1147–1161, 2003.
- [6] R.J. Vanderbei, D.N. Spergel, and N.J. Kasdin. Spiderweb Masks for High Contrast Imaging. *Astrophysical Journal*, 590:593–603, 2003.
- [7] R.J. Vanderbei, D.N. Spergel, and N.J. Kasdin. Circularly Symmetric Apodization via Starshaped Masks. *Astrophysical Journal*, 599:686–694, 2003.
- [8] R. J. Vanderbei, N. J. Kasdin, and D. N. Spergel. Checkerboard-Mask Coronagraphs for High-Contrast Imaging. *Astrophysical Journal*, 615(1):555, 2004.

**Thank You!**