Bilevel Derivative-Free Optimization and its Application to Robust Optimization

Andrew R. Conn, IBM Research

(joint work with L. N. Vicente, Univ. Coimbra)

ADVANCED METHODS AND PERSPECTIVES IN NONLINEAR OPTIMIZATION AND CONTROL,

Toulouse, February 3-5, 2010

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Growing sophistication of computer hardware and mathematical algorithms and software and a more competitive and complex world (which opens new possibilities for optimization).

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Growing sophistication of computer hardware and mathematical algorithms and software and a more competitive and complex world (which opens new possibilities for optimization).
- Function evaluations costly and noisy (one cannot trust derivatives or approximate them by finite differences).

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Growing sophistication of computer hardware and mathematical algorithms and software and a more competitive and complex world (which opens new possibilities for optimization).
- Function evaluations costly and noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available or owned by a company) — making automatic differentiation impossible to apply.

Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Growing sophistication of computer hardware and mathematical algorithms and software and a more competitive and complex world (which opens new possibilities for optimization).
- Function evaluations costly and noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available or owned by a company) — making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).

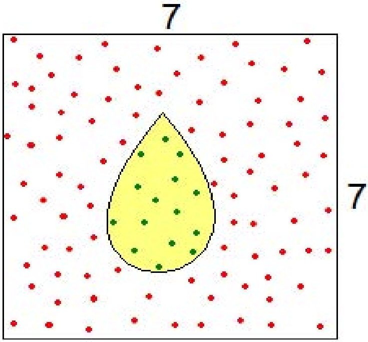
Why Derivative-Free Optimization?

Some of the reasons to apply derivative-free optimization are the following:

- Growing sophistication of computer hardware and mathematical algorithms and software and a more competitive and complex world (which opens new possibilities for optimization).
- Function evaluations costly and noisy (one cannot trust derivatives or approximate them by finite differences).
- Binary codes (source code not available or owned by a company) — making automatic differentiation impossible to apply.
- Legacy codes (written in the past and not maintained by the original authors).
- Lack of sophistication of the user (users need improvement but want to use something simple).

Simple Example with Unavailable Derivatives

Computation of areas of figures by random generation of points (the derivatives of the area function are clearly unavailable):



$$\text{Area} = 7 * 7 * \frac{15}{90+15} = 7$$

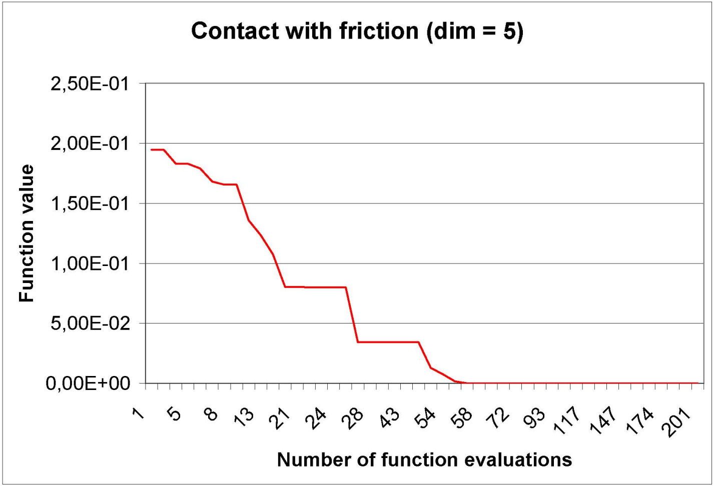
Examples of Problems where Derivatives are Unavailable

Many known applications:

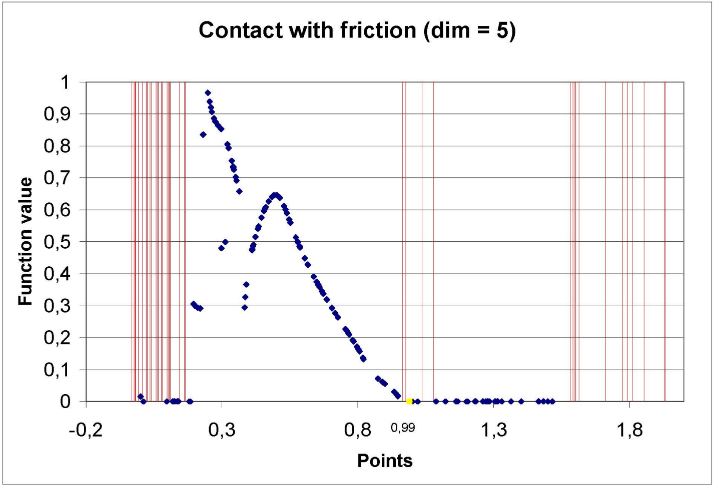
- Engineering design (many examples).
- Circuit design (tuning parameters of relatively small circuits using accurate simulation like PowerSpice).
- Molecular geometry optimization (minimization of the potential energy of clusters).
- Groundwater community problems.
- Medical image registration.
- Dynamic pricing.

- Tuning of algorithmic parameters.
- Automatic error analysis.

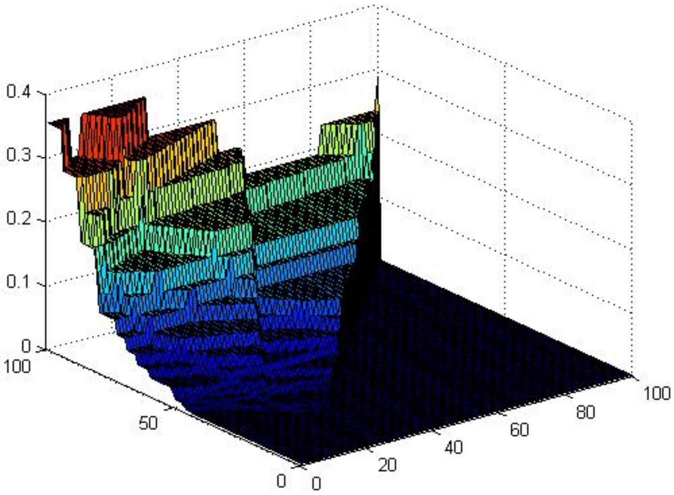
In DFO convergence/stopping is typically slow (per function evaluation):



The objective function not continuous or not well defined:



The objective function not continuous or not well defined:



What Can We Solve?

With current state-of-the-art DFO methods one can expect to successfully address problems where:

What Can We Solve?

With current state-of-the-art DFO methods one can expect to successfully address problems where:

- The evaluation of the function is expensive and/or computed with noise.

What Can We Solve?

With current state-of-the-art DFO methods one can expect to successfully address problems where:

- The evaluation of the function is expensive and/or computed with noise.
- The number of variables does not exceed, say, a few tens (in serial computation; maximum ≈ 200).

What Can We Solve?

With current state-of-the-art DFO methods one can expect to successfully address problems where:

- The evaluation of the function is expensive and/or computed with noise.
- The number of variables does not exceed, say, a few tens (in serial computation; maximum ≈ 200).
- The functions are not excessively nonsmooth.

What Can We Solve?

With current state-of-the-art DFO methods one can expect to successfully address problems where:

- The evaluation of the function is expensive and/or computed with noise.
- The number of variables does not exceed, say, a few tens (in serial computation; maximum ≈ 200).
- The functions are not excessively nonsmooth.
- Rapid asymptotic convergence is not of primary importance.

What Can We Solve?

With current state-of-the-art DFO methods one can expect to successfully address problems where:

- The evaluation of the function is expensive and/or computed with noise.
- The number of variables does not exceed, say, a few tens (in serial computation; maximum ≈ 200).
- The functions are not excessively nonsmooth.
- Rapid asymptotic convergence is not of primary importance.
- Only a few digits of accuracy are required.

What Can We Solve?

With current state-of-the-art DFO methods one can expect to successfully address problems where:

- The evaluation of the function is expensive and/or computed with noise.
- The number of variables does not exceed, say, a few tens (in serial computation; maximum ≈ 200).
- The functions are not excessively nonsmooth.
- Rapid asymptotic convergence is not of primary importance.
- Only a few digits of accuracy are required.

... making the linear algebra of the algorithms relatively inexpensive.

Illustration of Curse of Dimensionality

Number of points needed to build a complete/determined quadratic polynomial interpolant model:

n	10	20	50	100	200
$(n+1)(n+2)/2$	66	231	1326	5151	20301

Classes of Rigorous Algorithms (globally convergent)

Over-simplifying, all globally convergent DFO algorithms (model based or direct search) must:

Classes of Rigorous Algorithms (globally convergent)

Over-simplifying, all globally convergent DFO algorithms (model based or direct search) must:

- Guarantee some form of descent away from stationarity.

Classes of Rigorous Algorithms (globally convergent)

Over-simplifying, all globally convergent DFO algorithms (model based or direct search) must:

- Guarantee some form of descent away from stationarity.
- Guarantee some control of the geometry of the sample sets where the objective function is evaluated.

Classes of Rigorous Algorithms (globally convergent)

Over-simplifying, all globally convergent DFO algorithms (model based or direct search) must:

- Guarantee some form of descent away from stationarity.
- Guarantee some control of the geometry of the sample sets where the objective function is evaluated.
- Imply convergence of step size parameters to zero, indicating global convergence to a stationary point.

Classes of Rigorous Algorithms (globally convergent)

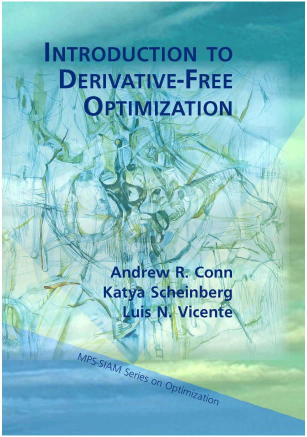
Over-simplifying, all globally convergent DFO algorithms (model based or direct search) must:

- Guarantee some form of descent away from stationarity.
- Guarantee some control of the geometry of the sample sets where the objective function is evaluated.
- Imply convergence of step size parameters to zero, indicating global convergence to a stationary point.

By global convergence, we mean convergence to some form of stationarity from arbitrary starting points.

The Book! (unashamed advertisement)

- A. R. Conn, K. Scheinberg, and L. N. Vicente, Introduction to Derivative-Free Optimization, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.



Trust-Region Methods for DFO (basics)

Trust-region methods for DFO typically:

Trust-Region Methods for DFO (basics)

Trust-region methods for DFO typically:

- Attempt to form quadratic models (by interpolation and using polynomials or radial basis functions)

$$m_k(x_k + \Delta x) \;=\; f(x_k) + g_k^\top \Delta x + \frac{1}{2}\Delta x^\top H_k \Delta x$$

  based on well-poised sample sets.

Trust-Region Methods for DFO (basics)

Trust-region methods for DFO typically:

- Attempt to form quadratic models (by interpolation and using polynomials or radial basis functions)

$$m_k(x_k + \Delta x) \;=\; f(x_k) + g_k^\top \Delta x + \frac{1}{2}\Delta x^\top H_k \Delta x$$

  based on well-poised sample sets.

$\Longrightarrow$ Well poisedness ensures fully linear or fully quadratic models.

Trust-Region Methods for DFO (basics)

Trust-region methods for DFO typically:

- Attempt to form quadratic models (by interpolation and using polynomials or radial basis functions)

$$m_k(x_k + \Delta x) \;=\; f(x_k) + g_k^\top \Delta x + \frac{1}{2}\Delta x^\top H_k \Delta x$$

based on well-poised sample sets.

$\Longrightarrow$ Well poisedness ensures fully linear or fully quadratic models.

- Calculate a step Δx_k by approximately solving the trust-region subproblem (TRS)

$$\min_{\Delta x \in B(x_k;\Delta_k)} \quad m_k(x_k + \Delta x).$$

Fully Linear Models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully linear if

Fully Linear Models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully linear if

- It is has Lipschitz continuous first derivatives.

Fully Linear Models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully linear if

- It is has Lipschitz continuous first derivatives.
- The following error bounds hold:

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\,\Delta \qquad \forall y \in B(x;\Delta)$$

and

$$|f(y) - m(y)| \leq \kappa_{ef}\,\Delta^2 \qquad \forall y \in B(x;\Delta).$$

Fully Linear Models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully linear if

- It is has Lipschitz continuous first derivatives.
- The following error bounds hold:

$$\|\nabla f(y) - \nabla m(y)\| \;\leq\; \kappa_{eg}\,\Delta \qquad \forall y \in B(x;\Delta)$$

and

$$|f(y) - m(y)| \;\leq\; \kappa_{ef}\,\Delta^2 \qquad \forall y \in B(x;\Delta).$$

For a class of fully-linear models, the (unknown) constants $\kappa_{ef}, \kappa_{eg} > 0$ must be independent of x and Δ.

Fully Quadratic Models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully quadratic if

Fully Quadratic Models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully quadratic if

- It is has Lipschitz continuous second derivatives.

Fully Quadratic Models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully quadratic if

- It is has Lipschitz continuous second derivatives.
- The following error bounds hold:

$$\|\nabla^2 f(y) - \nabla^2 m(y)\| \leq \kappa_{eh}\,\Delta \qquad \forall y \in B(x;\Delta)$$

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\,\Delta^2 \qquad \forall y \in B(x;\Delta)$$

and

$$|f(y) - m(y)| \leq \kappa_{ef}\,\Delta^3 \qquad \forall y \in B(x;\Delta).$$

Fully Quadratic Models

Given a point x and a trust-region radius Δ, a model $m(y)$ around x is called fully quadratic if

- It is has Lipschitz continuous second derivatives.
- The following error bounds hold:

$$\|\nabla^2 f(y) - \nabla^2 m(y)\| \leq \kappa_{eh}\, \Delta \qquad \forall y \in B(x;\Delta)$$

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\, \Delta^2 \qquad \forall y \in B(x;\Delta)$$

and

$$|f(y) - m(y)| \leq \kappa_{ef}\, \Delta^3 \qquad \forall y \in B(x;\Delta).$$

For a class of fully-quadratic models, the (unknown) constants $\kappa_{ef}, \kappa_{eg} > 0, \kappa_{eh} > 0$, must be independent of x and Δ.

Trust-Region Methods for DFO (basics : continued)

- Set x_{k+1} to $x_k + \Delta x_k$ (successful) or to x_k (unsuccessful) and update Δ_k depending on the value of

$$\rho_k = \frac{f(x_k) - f(x_k + \Delta x_k)}{m_k(x_k) - m_k(x_k + \Delta x_k)}.$$

Trust-Region Methods for DFO (basics : continued)

- Set x_{k+1} to $x_k + \Delta x_k$ (successful) or to x_k (unsuccessful) and update Δ_k depending on the value of

$$\rho_k = \frac{f(x_k) - f(x_k + \Delta x_k)}{m_k(x_k) - m_k(x_k + \Delta x_k)}.$$

- Attempt to accept steps based on simple decrease, i.e., if

$$\rho_k > 0 \iff f(x_k + \Delta x_k) < f(x_k).$$

Trust-Region Methods for DFO (main features)

- Reduce Δ_k only if ρ_k is small and the model is FL/FQ.

Trust-Region Methods for DFO (main features)

- Reduce Δ_k only if ρ_k is small and the model is FL/FQ.
- Accept new iterates based on simple decrease ($\rho_k > 0$) as long as the model is FL/FQ.

Trust-Region Methods for DFO (main features)

- Reduce Δ_k only if ρ_k is small and the model is FL/FQ.
- Accept new iterates based on simple decrease $(\rho_k > 0)$ as long as the model is FL/FQ.
- Allow for model-improving iterations (when ρ_k is not large enough and the model is not certifiably FL/FQ).

  $\Longrightarrow$ Do not reduce Δ_k.

Trust-Region Methods for DFO (main features)

- Reduce Δ_k only if ρ_k is small and the model is FL/FQ.

- Accept new iterates based on simple decrease $(\rho_k > 0)$ as long as the model is FL/FQ.

- Allow for model-improving iterations (when ρ_k is not large enough and the model is not certifiably FL/FQ).

  $\Longrightarrow$ Do not reduce Δ_k.

- Incorporate a criticality step (1st or 2nd order) when the 'stationarity' of the model is small.

  $\Longrightarrow$ Internal cycle of reductions of Δ_k.

Analysis of Trust-Region Methods (1st order)

Theorem (Book and SIOPT 2009 paper)

The trust-region radius converges to zero:

$$\Delta_k \longrightarrow 0.$$

Analysis of Trust-Region Methods (1st order)

Theorem (Book and SIOPT 2009 paper)

The trust-region radius converges to zero:

$$\Delta_k \longrightarrow 0.$$

Theorem (Book and SIOPT 2009 paper)

If f is bounded below and has Lipschitz continuous first derivatives then

$$\|\nabla f(x_k)\| \longrightarrow 0.$$

$\Longrightarrow$ True for simple decrease.

$\Longrightarrow$ Use of fully linear models when necessary.

Analysis of Trust-Region Methods (2nd order)

Theorem (Book and SIOPT 2009 paper)

The trust-region radius converges to zero ($\Delta_k \longrightarrow 0$).

Analysis of Trust-Region Methods (2nd order)

Theorem (Book and SIOPT 2009 paper)

The trust-region radius converges to zero ($\Delta_k \longrightarrow 0$).

Theorem (Book and SIOPT 2009 paper)

If f is bounded below and has Lipschitz continuous second derivatives then

$$\max\left\{\|\nabla f(x_k)\|, -\lambda_{min}[\nabla^2 f(x_k)]\right\} \longrightarrow 0.$$

$\Longrightarrow$ True for simple decrease (under a modification in the trust-region radius update).

$\Longrightarrow$ Use of fully quadratic models when necessary.

Inexact Function Values (dynamic accuracy)

Instead of $f(x)$ suppose we have $\bar{f}(x;\epsilon_x)$ and we can enforce

$$|f(x) - \bar{f}(x;\epsilon_x)| \;\leq\; \epsilon_x.$$

Inexact Function Values (dynamic accuracy)

Instead of $f(x)$ suppose we have $\bar{f}(x; \epsilon_x)$ and we can enforce

$$|f(x) - \bar{f}(x; \epsilon_x)| \ \leq \ \epsilon_x.$$

Suppose then $\max\{\epsilon_x, \epsilon_{x+s}\} \ \leq \ \eta_0' \, (m(x) - m(x+s))$.

Inexact Function Values (dynamic accuracy)

Instead of $f(x)$ suppose we have $\bar{f}(x; \epsilon_x)$ and we can enforce

$$|f(x) - \bar{f}(x; \epsilon_x)| \;\leq\; \epsilon_x.$$

Suppose then $\max\{\epsilon_x, \epsilon_{x+s}\} \;\leq\; \eta_0' \, (m(x) - m(x+s))$.

One knows (TR book, by Conn, Gould, and Toint, 2000) that if

$$\frac{\bar{f}(x; \epsilon_x) - \bar{f}(x+s; \epsilon_{x+s})}{m(x) - m(x+s)} \;\geq\; \eta_0$$

Inexact Function Values (dynamic accuracy)

Instead of $f(x)$ suppose we have $\bar{f}(x;\epsilon_x)$ and we can enforce

$$|f(x) - \bar{f}(x;\epsilon_x)| \;\leq\; \epsilon_x.$$

Suppose then $\max\{\epsilon_x, \epsilon_{x+s}\} \;\leq\; \eta_0' \,(m(x) - m(x+s))$.

One knows (TR book, by Conn, Gould, and Toint, 2000) that if

$$\frac{\bar{f}(x;\epsilon_x) - \bar{f}(x+s;\epsilon_{x+s})}{m(x) - m(x+s)} \;\geq\; \eta_0$$

then

$$\frac{f(x) - f(x+s)}{m(x) - m(x+s)} \;\geq\; \eta_0' - 2\eta_0 \;>\; 0,$$

with $0 < \eta_0 < \eta_0'/2$ and $\eta_0 < 1$.

Polynomial Models

Given a sample set $Y = \{y^0, y^1, \ldots, y^p\}$, a polynomial basis ϕ, and a polynomial model $m(y) = \alpha^\top \phi(y)$, the interpolating conditions are the following system of linear equations:

$$M(\phi, Y)\alpha \; = \; f(Y),$$

Polynomial Models

Given a sample set $Y = \{y^0, y^1, \ldots, y^p\}$, a polynomial basis ϕ, and a polynomial model $m(y) = \alpha^\top \phi(y)$, the interpolating conditions are the following system of linear equations:

$$M(\phi, Y)\alpha \; = \; f(Y),$$

where

$$M(\phi, Y) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_p(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_p(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_p(y^p) \end{bmatrix} \quad f(Y) = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^p) \end{bmatrix}.$$

Polynomial Models

Given a sample set $Y = \{y^0, y^1, \ldots, y^p\}$, a polynomial basis ϕ, and a polynomial model $m(y) = \alpha^\top \phi(y)$, the interpolating conditions are the following system of linear equations:

$$M(\phi, Y)\alpha = f(Y),$$

where

$$M(\phi, Y) = \begin{bmatrix} \phi_0(y^0) & \phi_1(y^0) & \cdots & \phi_p(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \cdots & \phi_p(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \cdots & \phi_p(y^p) \end{bmatrix} \quad f(Y) = \begin{bmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^p) \end{bmatrix}.$$

We use the natural basis of monomials, which in 2D is

$$\phi = \{1, x_1, x_2, x_1^2/2, x_2^2/2, x_1 x_2\}.$$

Example of the Interpolation Matrix (underdetermined)

Let us focus on the underdetermined case where (# points) $<$ (# basis components).

Example of the Interpolation Matrix (underdetermined)

Let us focus on the underdetermined case where (# points) $<$ (# basis components).

For instance, when $n = d = 2$, $p = 3$, and

$$\phi \;=\; \{1, x_1, x_2, x_1^2/2, x_2^2/2, x_1 x_2\},$$

the matrix $M(\phi, Y)$ becomes

$$\left[\begin{array}{cccccc} 1 & y_1^0 & y_2^0 & (y_1^0)^2/2 & y_1^0 y_2^0 & (y_2^0)^2/2 \\ 1 & y_1^1 & y_2^1 & (y_1^1)^2/2 & y_1^1 y_2^1 & (y_2^1)^2/2 \\ 1 & y_1^2 & y_2^2 & (y_1^2)^2/2 & y_1^2 y_2^2 & (y_2^2)^2/2 \\ 1 & y_1^3 & y_2^3 & (y_1^3)^2/2 & y_1^3 y_2^3 & (y_2^3)^2/2 \end{array}\right].$$

Underdetermined Polynomial Models

Consider a underdetermined quadratic polynomial model

$$m(y) \;=\; c + g^\top y + \frac{1}{2} y^\top H y$$

built with less than $(n+1)(n+2)/2$ points.

Underdetermined Polynomial Models

Consider a underdetermined quadratic polynomial model

$$m(y) \ = \ c + g^\top y + \frac{1}{2} y^\top H y$$

built with less than $(n+1)(n+2)/2$ points.

Theorem (Book)

If Y is Λ_L–poised for linear interpolation/regression then

$$\|\nabla f(y) - \nabla m(y)\| \ \leq \ \Lambda_L \left[C_f + \|H\| \right] \Delta \qquad \forall y \in B(x; \Delta).$$

Underdetermined Polynomial Models

Consider a underdetermined quadratic polynomial model

$$m(y) \;=\; c + g^\top y + \frac{1}{2} y^\top H y$$

built with less than $(n+1)(n+2)/2$ points.

Theorem (Book)

If Y is Λ_L–poised for linear interpolation/regression then

$$\|\nabla f(y) - \nabla m(y)\| \;\leq\; \Lambda_L \left[C_f + \|H\| \right] \Delta \qquad \forall y \in B(x;\Delta).$$

$\Longrightarrow$ Linear Λ_L–poisedness is equivalent to $\|M(\phi_L, Y_{scaled})^\dagger\| \leq \Lambda_L$.

Underdetermined Polynomial Models (continued)

Again,

$$\|\nabla f(y) - \nabla m(y)\| \;\leq\; \Lambda_L \left[C_f + \|H\|\right] \Delta \qquad \forall y \in B(x;\Delta).$$

Again,

$$\|\nabla f(y) - \nabla m(y)\| \;\leq\; \Lambda_L \left[C_f + \|H\|\right] \Delta \qquad \forall y \in B(x;\Delta).$$

Q: What should we do?

Underdetermined Polynomial Models (continued)

Again,

$$\|\nabla f(y) - \nabla m(y)\| \;\leq\; \Lambda_L \left[C_f + \|H\|\right] \Delta \qquad \forall y \in B(x;\Delta).$$

Q: What should we do?

A: One should build models by minimizing the norm of H.

Minimum Frobenius Norm Models

Recall the sample set $Y = \{y^0, y^1, \ldots, y^p\}$ and the quadratic model

$$m(y) \; = \; c + g^\top y + \frac{1}{2} y^\top H y \; = \; \alpha_L^\top \phi_L(x) + \alpha_Q^\top \phi_Q(x).$$

Minimum Frobenius Norm Models

Recall the sample set $Y = \{y^0, y^1, \ldots, y^p\}$ and the quadratic model

$$m(y) \ = \ c + g^\top y + \frac{1}{2} y^\top H y \ = \ \alpha_L^\top \phi_L(x) + \alpha_Q^\top \phi_Q(x).$$

MFN models can be built by minimizing the entries of the Hessian (in the Frobenius norm) subject to the interpolation conditions:

$$\begin{array}{ll} \min & \frac{1}{4}\|H\|_F^2 \\ \text{s.t.} & c + g^\top (y^i) + \frac{1}{2}(y^i)^\top H (y^i) \ = \ f(y^i), \quad i = 0, \ldots, p, \end{array}$$

Minimum Frobenius Norm Models

Recall the sample set $Y = \{y^0, y^1, \ldots, y^p\}$ and the quadratic model

$$m(y) \;=\; c + g^\top y + \frac{1}{2} y^\top H y \;=\; \alpha_L^\top \phi_L(x) + \alpha_Q^\top \phi_Q(x).$$

MFN models can be built by minimizing the entries of the Hessian (in the Frobenius norm) subject to the interpolation conditions:

$$\begin{array}{ll} \min & \frac{1}{4}\|H\|_F^2 \\ \text{s.t.} & c + g^\top (y^i) + \frac{1}{2}(y^i)^\top H (y^i) \;=\; f(y^i), \quad i = 0, \ldots, p, \end{array}$$

or, equivalently,

$$\begin{array}{ll} \min & \frac{1}{2}\|\alpha_Q\|^2 \\ \text{s.t.} & M(\phi, Y)\alpha \;=\; f(Y). \end{array}$$

The solution of this QP problem requires a linear solve with:

$$F(\phi, Y) \;=\; \left[\begin{array}{cc} M(\phi_Q, Y) M(\phi_Q, Y)^\top & M(\phi_L, Y) \\ M(\phi_L, Y)^\top & 0 \end{array} \right],$$

where

$$M(\phi, Y) \;=\; \left[\begin{array}{cc} M(\phi_L, Y) & M(\phi_Q, Y) \end{array} \right].$$

Minimum Frobenius Norm Models (continued)

The solution of this QP problem requires a linear solve with:

$$F(\phi, Y) \;=\; \left[\begin{array}{cc} M(\phi_Q, Y) M(\phi_Q, Y)^\top & M(\phi_L, Y) \\ M(\phi_L, Y)^\top & 0 \end{array} \right],$$

where

$$M(\phi, Y) \;=\; \left[\begin{array}{cc} M(\phi_L, Y) & M(\phi_Q, Y) \end{array} \right].$$

$\Longrightarrow$ Λ_F–poisedness in the minimum Frobenius norm is equivalent to:

$$\|F(\phi, Y_{scaled})^{-1}\| \;\leq\; \Lambda_F.$$

Minimum Frobenius Norm Models (continued)

Theorem (Book)

If Y is Λ_F–poised in the minimum Frobenius norm sense then

$$\|H\| \;\leq\; C_n C_f \Lambda_F,$$

where H is, again, the Hessian of the model.

Minimum Frobenius Norm Models (continued)

Theorem (Book)

If Y is Λ_F–poised in the minimum Frobenius norm sense then

$$\|H\| \;\leq\; C_n C_f \Lambda_F,$$

where H is, again, the Hessian of the model.

Putting the two theorems together yields:

$$\|\nabla f(y) - \nabla m(y)\| \;\leq\; \Lambda_L \left[C_f + C_n C_f \Lambda_F\right] \Delta \quad \forall y \in B(x;\Delta).$$

Minimum Frobenius Norm Models (continued)

Theorem (Book)

If Y is Λ_F–poised in the minimum Frobenius norm sense then

$$\|H\| \;\leq\; C_n C_f \Lambda_F,$$

where H is, again, the Hessian of the model.

Putting the two theorems together yields:

$$\|\nabla f(y) - \nabla m(y)\| \;\leq\; \Lambda_L \left[C_f + C_n C_f \Lambda_F\right] \Delta \quad \forall y \in B(x;\Delta).$$

$\Longrightarrow$ MFN models are fully linear.

The Bilevel Programming Problem

$$\min_{(x^u, x^\ell) \in \mathbb{R}^{n^u \times n^\ell}} \quad f^u(x^u, x^\ell)$$

$$\begin{aligned} \text{subject to} \quad & c_i^u(x^u, x^\ell) = 0 & i \in \mathcal{E}^u \\ & c_i^u(x^u, x^\ell) \geq 0 & i \in \mathcal{I}^u \end{aligned}$$

The Bilevel Programming Problem

$$\min_{(x^u, x^\ell) \in \mathbb{R}^{n^u \times n^\ell}} \quad f^u(x^u, x^\ell)$$

$$\begin{aligned} \text{subject to} \quad & c_i^u(x^u, x^\ell) = 0 & i \in \mathcal{E}^u \\ & c_i^u(x^u, x^\ell) \geq 0 & i \in \mathcal{I}^u \end{aligned}$$

where x^ℓ is the

$$\arg\min_{z^\ell \in \mathbb{R}^{n^\ell}} \quad f^\ell(x^u, z^\ell)$$

$$\begin{aligned} \text{subject to} \quad & c_i^\ell(x^u, z^\ell) = 0 & i \in \mathcal{E}^\ell \\ & c_i^\ell(x^u, z^\ell) \geq 0 & i \in \mathcal{I}^\ell. \end{aligned}$$

Why Bilevel? Why is it Hard?

- Particular but important class of problems.

Why Bilevel? Why is it Hard?

- Particular but important class of problems.
- Unfortunately very difficult — even with available derivatives.

Why Bilevel? Why is it Hard?

- Particular but important class of problems.
- Unfortunately very difficult — even with available derivatives.
- Even in the purely linear case it is non-convex.

Why Bilevel? Why is it Hard?

- Particular but important class of problems.
- Unfortunately very difficult — even with available derivatives.
- Even in the purely linear case it is non-convex.
- Due to upper level constraints, the feasible region might be disconnected.

Why Bilevel? Why is it Hard?

- Particular but important class of problems.
- Unfortunately very difficult — even with available derivatives.
- Even in the purely linear case it is non-convex.
- Due to upper level constraints, the feasible region might be disconnected.

- Applications arise, e.g., in governmental, agricultural, and environmental problems.

Why Bilevel? Why is it Hard?

- Particular but important class of problems.
- Unfortunately very difficult — even with available derivatives.
- Even in the purely linear case it is non-convex.
- Due to upper level constraints, the feasible region might be disconnected.

- Applications arise, e.g., in governmental, agricultural, and environmental problems.
- Applications also appear in engineering (e.g., robust optimization).

The Bilevel Problem Without Additional Constraints

We will ignore the constraints for each level:

$$\min_{(x^u, x^\ell) \in \mathbb{R}^{n^u \times n^\ell}} \quad f^u(x^u, x^\ell)$$

The Bilevel Problem Without Additional Constraints

We will ignore the constraints for each level:

$$\min_{(x^u, x^\ell) \in \mathbb{R}^{n^u \times n^\ell}} \quad f^u(x^u, x^\ell)$$

where x^ℓ is the

$$\operatorname*{arg\,min}_{z^\ell \in \mathbb{R}^{n^\ell}} \quad f^\ell(x^u, z^\ell).$$

The Reduced Formulation

If we define the set of lower level minimizers (assumed a singleton) as

$$x^\ell(x^u) = \arg\min \left\{ f^\ell(x^u, x^\ell) : \; x^\ell \in \mathbb{R}^{n^\ell} \right\},$$

The Reduced Formulation

If we define the set of lower level minimizers (assumed a singleton) as

$$x^\ell(x^u) = \arg\min \left\{ f^\ell(x^u, x^\ell) : \ x^\ell \in \mathbb{R}^{n^\ell} \right\},$$

one can rewrite the bilevel problem in the upper level variables (so-called reduced formulation)

$$\min_{x^u \in \mathbb{R}^{n^u}} \ f^u(x^u, x^\ell(x^u)).$$

The Reduced Formulation

If we define the set of lower level minimizers (assumed a singleton) as

$$x^\ell(x^u) = \arg\min \left\{ f^\ell(x^u, x^\ell) : \, x^\ell \in \mathbb{R}^{n^\ell} \right\},$$

one can rewrite the bilevel problem in the upper level variables (so-called reduced formulation)

$$\min_{x^u \in \mathbb{R}^{n^u}} \; f^u(x^u, x^\ell(x^u)).$$

We will call $f^u(x^u, x^\ell(x^u))$ the reduced upper level function.

The Lower Level Solution

One could think of the lower level problem as a constraint of the upper level problem.

The Lower Level Solution

One could think of the lower level problem as a constraint of the upper level problem.

Using the first-order conditions, that constraint looks like

$$\nabla_z f^{\ell}(x^u, z^*) = 0.$$

The Lower Level Solution

One could think of the lower level problem as a constraint of the upper level problem.

Using the first-order conditions, that constraint looks like

$$\nabla_z f^\ell(x^u, z^*) = 0.$$

So, it seems reasonable to suggest that one needs to satisfy the first-order conditions to first-order ...

The Lower Level Solution

One could think of the lower level problem as a constraint of the upper level problem.

Using the first-order conditions, that constraint looks like

$$\nabla_z f^\ell(x^u, z^*) = 0.$$

So, it seems reasonable to suggest that one needs to satisfy the first-order conditions to first-order ...

... which suggests solving the lower level problem using fully quadratic models.

Our DFO Approach for Bilevel Optimization

Considers the reduced formulation.

Our DFO Approach for Bilevel Optimization

Considers the reduced formulation.

Applies a trust-region interpolation-based method to both upper and lower level problems.

Our DFO Approach for Bilevel Optimization

Considers the reduced formulation.

Applies a trust-region interpolation-based method to both upper and lower level problems.

Explores the theory of TR methods and the structure of the lower level problem to gain efficiency:

Our DFO Approach for Bilevel Optimization

Considers the reduced formulation.

Applies a trust-region interpolation-based method to both upper and lower level problems.

Explores the theory of TR methods and the structure of the lower level problem to gain efficiency:

1. solving the lower level problem inexactly,

Our DFO Approach for Bilevel Optimization

Considers the reduced formulation.

Applies a trust-region interpolation-based method to both upper and lower level problems.

Explores the theory of TR methods and the structure of the lower level problem to gain efficiency:

1. solving the lower level problem inexactly,
2. reusing previous (upper level perturbed) evaluated points,

Our DFO Approach for Bilevel Optimization

Considers the reduced formulation.

Applies a trust-region interpolation-based method to both upper and lower level problems.

Explores the theory of TR methods and the structure of the lower level problem to gain efficiency:

1. solving the lower level problem inexactly,
2. reusing previous (upper level perturbed) evaluated points,
3. not ignoring dynamic accuracy in the TR methods.

Our Trust-Region Interpolation-based Method

We follow Fasano, Morales, and Nocedal, 2009: When the iteration is successful, the new point is always brought to the sample set.

Our Trust-Region Interpolation-based Method

We follow Fasano, Morales, and Nocedal, 2009: When the iteration is successful, the new point is always brought to the sample set.

Differently, we discard the sample point farthest away from the new iterate.

Our Trust-Region Interpolation-based Method

We follow Fasano, Morales, and Nocedal, 2009: When the iteration is successful, the new point is always brought to the sample set.

Differently, we discard the sample point farthest away from the new iterate.

Differently, we start with less than $p_{\max} = (n+1)(n+2)/2$ points and use MFN models.

Our Trust-Region Interpolation-based Method

We follow Fasano, Morales, and Nocedal, 2009: When the iteration is successful, the new point is always brought to the sample set.

Differently, we discard the sample point farthest away from the new iterate.

Differently, we start with less than $p_{\max} = (n+1)(n+2)/2$ points and use MFN models.

Thus, until $|$sample set$|$ reaches $p_{\max}$, we never discard points from the sample set and always add new trial points independently of being accepted or not as new iterates.

Our Trust-Region Interpolation-based Method (continued)

Differently also, we discard points that are too far from the current iterate when the trust radius becomes small — can be seen as a form of criticality step.

Our Trust-Region Interpolation-based Method (continued)

Differently also, we discard points that are too far from the current iterate when the trust radius becomes small — can be seen as a form of criticality step.

Thus, |sample set| might get below $p_{\min} = n + 1$ (the number required to build fully linear models).

Differently also, we discard points that are too far from the current iterate when the trust radius becomes small — can be seen as a form of criticality step.

Thus, $|$sample set$|$ might get below $p_{\min} = n + 1$ (the number required to build fully linear models).

In such situations, we never reduce the trust radius.

Inexact Solution of the Lower Level Problem

Under reasonable assumptions, one can prove

$$|f^u(x^u, x^\ell(x^u)) - f^u(x^u, x^\ell_{dfo}(x^u))| \leq \mathcal{O}(\|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\|) + \mathcal{O}((\Delta^\ell)^2).$$

Inexact Solution of the Lower Level Problem

Under reasonable assumptions, one can prove

$$|f^u(x^u, x^\ell(x^u)) - f^u(x^u, x^\ell_{dfo}(x^u))| \leq \mathcal{O}(\|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\|) + \mathcal{O}((\Delta^\ell)^2).$$

Thus, if

$$\|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\| = \mathcal{O}((\Delta^u)^2) \quad \text{and} \quad \Delta^\ell = \mathcal{O}(\Delta^u),$$

Inexact Solution of the Lower Level Problem

Under reasonable assumptions, one can prove

$$|f^u(x^u, x^\ell(x^u)) - f^u(x^u, x^\ell_{dfo}(x^u))| \;\leq\; \mathcal{O}(\|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\|) + \mathcal{O}((\Delta^\ell)^2).$$

Thus, if

$$\|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\| \;=\; \mathcal{O}((\Delta^u)^2) \quad \text{and} \quad \Delta^\ell \;=\; \mathcal{O}(\Delta^u),$$

then

$$|f^u(x^u, x^\ell(x^u)) - f^u(x^u, x^\ell_{dfo}(x^u))| \;=\; \mathcal{O}((\Delta^u)^2),$$

Inexact Solution of the Lower Level Problem

Under reasonable assumptions, one can prove

$$
\begin{aligned}
|f^u(x^u, x^\ell(x^u)) - f^u(x^u, x^\ell_{dfo}(x^u))| \;\; \leq \;\; & \mathcal{O}(\|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\|) \\
& + \mathcal{O}((\Delta^\ell)^2).
\end{aligned}
$$

Thus, if

$$
\|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\| \; = \; \mathcal{O}((\Delta^u)^2) \quad \text{and} \quad \Delta^\ell \; = \; \mathcal{O}(\Delta^u),
$$

then

$$
|f^u(x^u, x^\ell(x^u)) - f^u(x^u, x^\ell_{dfo}(x^u))| \; = \; \mathcal{O}((\Delta^u)^2),
$$

and one can prove that the upper level model stays fully linear.

Dynamic Accuracy Requirements

One way to approximately enforce the dynamic accuracy requirement is to consider only

$$\epsilon_{x^u+s^u} \leq \eta_0'(m^u(x^u) - m^u(x^u + s^u))$$

which is satisfied if

Dynamic Accuracy Requirements

One way to approximately enforce the dynamic accuracy requirement is to consider only

$$\epsilon_{x^u+s^u} \leq \eta_0'(m^u(x^u) - m^u(x^u+s^u))$$

which is satisfied if

$$\|\nabla_\ell m^\ell(x^u+s^u, x^\ell_{dfo}(x^u+s^u))\| \; = \; \mathcal{O}\left(\min(\|s^u\|^2, \|s^u\|\|g^u\|)\right)$$

Dynamic Accuracy Requirements

One way to approximately enforce the dynamic accuracy requirement is to consider only

$$\epsilon_{x^u+s^u} \leq \eta_0'(m^u(x^u) - m^u(x^u + s^u))$$

which is satisfied if

$$\|\nabla_\ell m^\ell(x^u + s^u, x^\ell_{dfo}(x^u + s^u))\| \ = \ \mathcal{O}\left(\min(\|s^u\|^2, \|s^u\|\|g^u\|)\right)$$

and

$$\Delta^\ell \ = \ \mathcal{O}\left(\sqrt{\min(\|s^u\|^2, \|s^u\|\|g^u\|)}\right).$$

Reusing Previous (Upper Level Perturbed) Evaluated Points

Since

$$|f^\ell(x^u, x^\ell) - f^\ell(x^u_{pert}, x^\ell)| \;\leq\; \mathcal{O}(\|x^u - x^u_{pert}\|),$$

Reusing Previous (Upper Level Perturbed) Evaluated Points

Since

$$|f^\ell(x^u, x^\ell) - f^\ell(x^u_{pert}, x^\ell)| \;\leq\; \mathcal{O}(\|x^u - x^u_{pert}\|),$$

if

$$\|x^u - x^u_{pert}\| \;=\; \mathcal{O}((\Delta^\ell)^3),$$

Since

$$|f^\ell(x^u, x^\ell) - f^\ell(x^u_{pert}, x^\ell)| \;\leq\; \mathcal{O}(\|x^u - x^u_{pert}\|),$$

if

$$\|x^u - x^u_{pert}\| \;=\; \mathcal{O}((\Delta^\ell)^3),$$

then

$$|f^\ell(x^u, x^\ell) - f^\ell(x^u_{pert}, x^\ell)| \;=\; \mathcal{O}((\Delta^\ell)^3).$$

Since

$$|f^\ell(x^u, x^\ell) - f^\ell(x^u_{pert}, x^\ell)| \;\leq\; \mathcal{O}(\|x^u - x^u_{pert}\|),$$

if

$$\|x^u - x^u_{pert}\| \;=\; \mathcal{O}((\Delta^\ell)^3),$$

then

$$|f^\ell(x^u, x^\ell) - f^\ell(x^u_{pert}, x^\ell)| \;=\; \mathcal{O}((\Delta^\ell)^3).$$

This provide us a criterion to decide whether to accept previously evaluated points in the building of the lower level model.

Matlab Code

We developed a relatively sophisticated Matlab implementation along the lines described above.

Matlab Code

We developed a relatively sophisticated Matlab implementation along the lines described above.

The code handles bilevel problems with any type of linear constraints except upper level constraints on the lower level variables.

Matlab Code

We developed a relatively sophisticated Matlab implementation along the lines described above.

The code handles bilevel problems with any type of linear constraints except upper level constraints on the lower level variables.

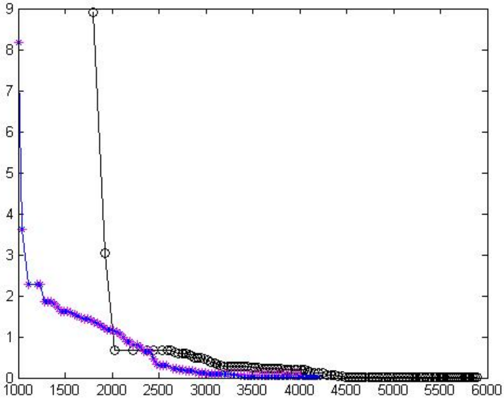
Another feature not described is a warm start procedure for initialization of lower level variables by forming a linear model of $x^\ell(x^u)$.

Quadratic/Quartic 5×5 Example

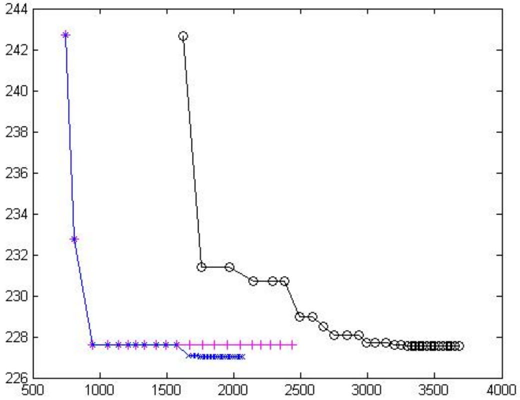
Black: basic version

Red: inexact lower level

Blue: inexact lower level & reuse of points

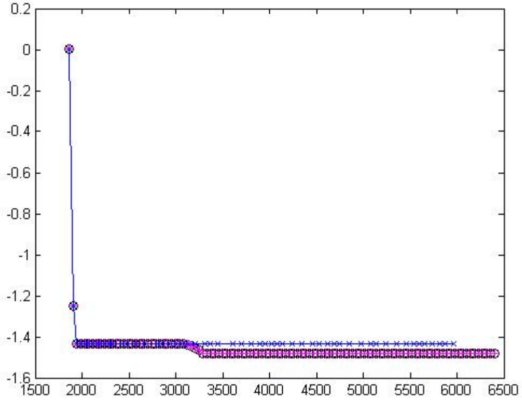


Black: basic version
Red: inexact lower level
Blue: inexact lower level & reuse of points



Cubic/Quadratic 20×20 example, with linear constraints

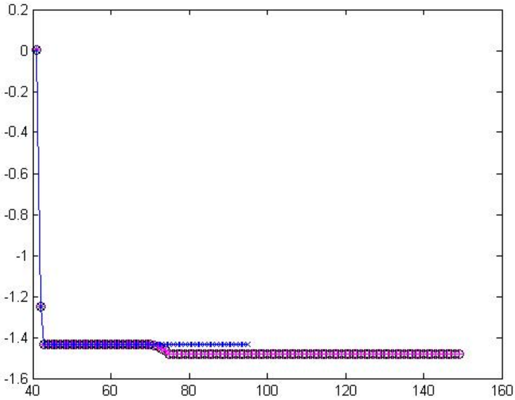
Black: basic version
Red: inexact lower level
Blue: inexact lower level & reuse of points



Cubic/Quadratic 20×20 example, with linear constraints

Black: basic version Red: inexact lower level

Blue: inexact lower level & reuse of points



Same $f^u(x^u, x^\ell_{dfo}(x^u))$ values (but now as a function of the # ul evaluations).

Robust Optimization

In many real-world optimization problems, data is uncertain and a representation is calculated by some form of estimation.

Robust Optimization

In many real-world optimization problems, data is uncertain and a representation is calculated by some form of estimation.

In robust optimization, immunization against data uncertainty is made by letting the uncertain parameters p vary in uncertainty sets $\mathcal{P}$ ...

Robust Optimization

In many real-world optimization problems, data is uncertain and a representation is calculated by some form of estimation.

In robust optimization, immunization against data uncertainty is made by letting the uncertain parameters p vary in uncertainty sets $\mathcal{P}$ ...

... and by looking for a safe, worst case scenario:

$$\min_{x \in \mathbb{R}^n} \max_{p \in \mathcal{P}} f(x, p).$$

Robust Optimization

In many real-world optimization problems, data is uncertain and a representation is calculated by some form of estimation.

In robust optimization, immunization against data uncertainty is made by letting the uncertain parameters p vary in uncertainty sets $\mathcal{P}$ ...

... and by looking for a safe, worst case scenario:

$$\min_{x \in \mathbb{R}^n} \max_{p \in \mathcal{P}} f(x, p).$$

Robust optimization also provides a tool for dealing with variables for which the optimal values must be later implemented.

Robust Optimization as Bilevel

This problem can be reformulated as a bilevel optimization problem of the form

$$\min_{(x,p)\in\mathbb{R}^n\times\mathcal{P}} \quad f(x,p)$$

This problem can be reformulated as a bilevel optimization problem of the form

$$\min_{(x,p)\in\mathbb{R}^n\times\mathcal{P}} \quad f(x,p)$$

where p is the

$$\arg\min_{z\in\mathcal{P}} \quad -f(x,z).$$

Small Robust Example

We tested our algorithm in the example reported in Bertsimas, Nohadani, Teo, 2010.

Small Robust Example

We tested our algorithm in the example reported in Bertsimas, Nohadani, Teo, 2010.

The robust function is $f(x, p) = g(x + p)$, where $x, p \in \mathbb{R}^2$ and

$$\begin{aligned} g(x) &= 2x_1^6 - 12.2x_1^5 + 21.2x_1^4 - 6.4x_1^3 - 4.7x_1^2 + 6.2x_1 \\ &\quad + x_2^6 - 11x_2^5 + 43.3x_2^4 - 74.8x_2^3 + 56.9x_2^2 - 10x_2 \\ &\quad - 0.1x_1^2x_2^2 + 0.4x_1^2x_2 + 0.4x_2^2x_1 - 4.1x_1x_2. \end{aligned}$$

Small Robust Example

We tested our algorithm in the example reported in Bertsimas, Nohadani, Teo, 2010.

The robust function is $f(x,p) = g(x+p)$, where $x, p \in \mathbb{R}^2$ and

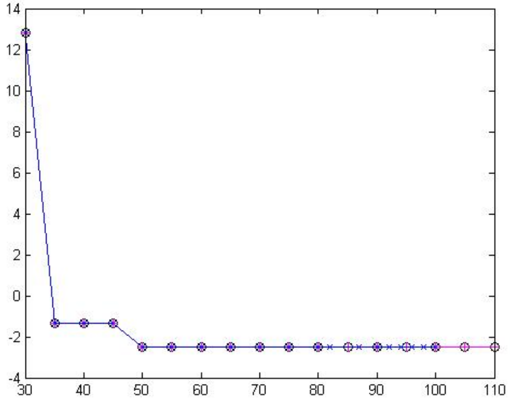
$$\begin{aligned} g(x) \;=\; & 2x_1^6 - 12.2x_1^5 + 21.2x_1^4 - 6.4x_1^3 - 4.7x_1^2 + 6.2x_1 \\ & + x_2^6 - 11x_2^5 + 43.3x_2^4 - 74.8x_2^3 + 56.9x_2^2 - 10x_2 \\ & - 0.1x_1^2x_2^2 + 0.4x_1^2x_2 + 0.4x_2^2x_1 - 4.1x_1x_2. \end{aligned}$$

The problem has one lower level constraint of the form $\|p\| \leq 0.5$ describing implementation errors:

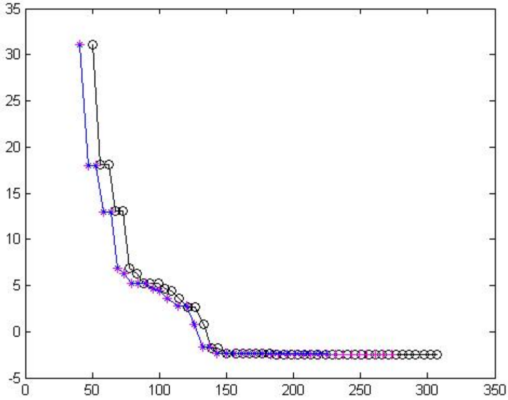
$$\begin{aligned} \min_{x\in\mathbb{R}^2, p\in\mathbb{R}^2} \quad & g(x+p) \\ \text{s.t.} \quad & p \in \arg\min \left\{ -g(x+p) : \; p \in \mathbb{R}^2, \|p\| \leq 0.5 \right\}. \end{aligned}$$

Bertsimas et al. Example (initial point A)

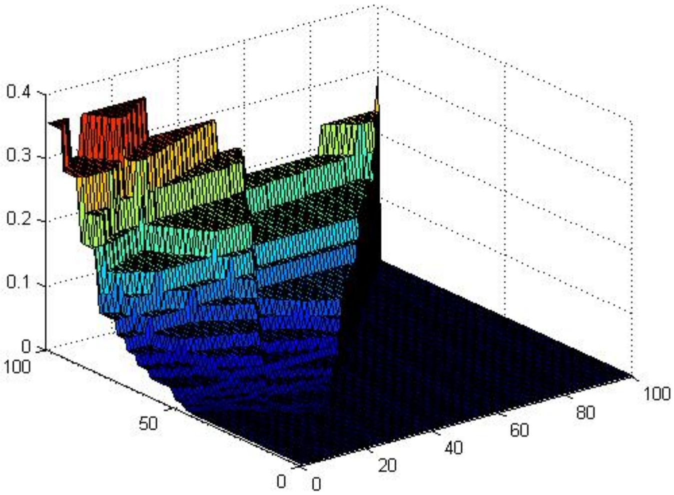
Black: basic version
Red: inexact lower level
Blue: inexact lower level & reuse of points



Black: basic version
Red: inexact lower level
Blue: inexact lower level & reuse of points



We are currently solving robust portfolio problems involving the Omega function.



What is the Omega Function?

Let the random variable R model the return for some financial instrument.

What is the Omega Function?

Let the random variable R model the return for some financial instrument. Consider a return level/threshold L.

What is the Omega Function?

Let the random variable R model the return for some financial instrument. Consider a return level/threshold L.

The Omega function is the ratio of the weighted gains (above L) over the weighted losses (below L):

$$\Omega(R) = \frac{\int_{L}^{L_{max}} \mathbb{P}(R \geq r)\, dr}{\int_{L_{min}}^{L} \mathbb{P}(R \leq r)\, dr}.$$

What is the Omega Function?

Let the random variable R model the return for some financial instrument. Consider a return level/threshold L.
The Omega function is the ratio of the weighted gains (above L) over the weighted losses (below L):

$$\Omega(R) = \frac{\int_L^{L_{max}} \mathbb{P}(R \geq r)\, dr}{\int_{L_{min}}^{L} \mathbb{P}(R \leq r)\, dr}.$$

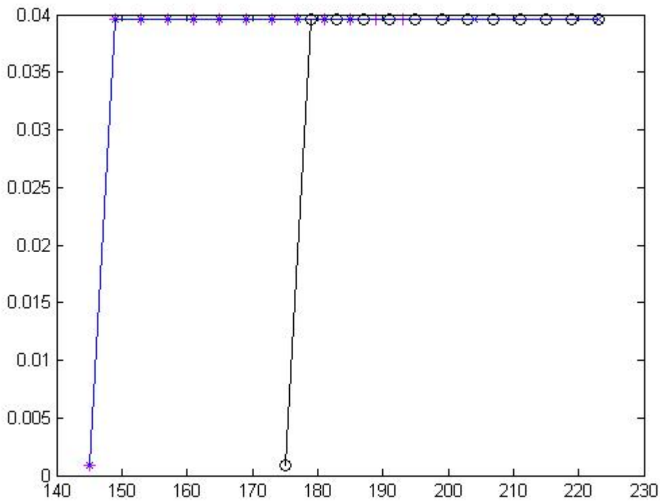
For portfolio optimization, one considers $\Omega(x_1 R_1 + \cdots + x_n R_n)$ and minimize over $x_1 + \cdots + x_n = 1$ and $x_1, \ldots, x_n \geq 0$
and the single threshold parameter L is allowed to vary for robustness in $[0, 0.04]$.

$$\min_{(x^u, x^\ell) \in \mathbb{R}^7 \times [0, 0.04]} \quad -\Omega(x^u; x^\ell)$$
$$\text{s.t.} \quad x^\ell \in \arg\min \left\{ \Omega(x^u; z^\ell) : \; z^\ell \in [0, 0.04] \right\}.$$

Maximizing the Omega Function

8 assets = 7 upper level variables

1 return level = 1 lower level variable / robust variable



References

- A. R. Conn and L. N. Vicente,
  Bilevel derivative-free optimization and its application to robust optimization, in preparation.

- A. R. Conn, K. Scheinberg, and L. N. Vicente,
  Global convergence of general derivative-free trust-region algorithms to first and second order critical points,
  SIAM J. on Optimization, Vol. 20, No. 1, pp. 387 — 415, 2009.

- G. Fasano, J. L. Morales, and J. Nocedal,
  On the Geometry Phase in Model-Based Algorithms for Derivative-Free Optimization,
  Optimization Methods and Software, Vol. 24, No. 1, pp. 145 — 154, 2008.

- A. R. Conn, N. I. M. Gould, and Ph. L. Toint,
  Trust-Region Methods,
  MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2000.

- A. R. Conn, K. Scheinberg, and L. N. Vicente, Introduction to Derivative-Free Optimization, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.