Generating Set Search Methods for Nonlinear Optimization

Robert Michael Lewis and Virginia Torczon, College of William & Mary

Collaborators:

- Tammy Kolda, Sandia National Laboratories, Livermore, California
- Anne Shepherd, College of William & Mary
- Chris Siefert, Sandia National Laboratories, Albuquerque, New Mexico
- Michael Trosset, Indiana University

The general nonlinear optimization/nonlinear programming problem

$$
\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & x \in \mathcal{S} \subseteq \mathbb{R}^n.
\end{array}
$$

Categorization for nonlinear programming

Unconstrained: $\mathcal{S} = \mathbb{R}^n$.

Bound constrained: $\mathcal{S} = \{x \mid \ell \leq x \leq u\}$, where $\ell, u \in \mathbb{R}^n$.

Linearly constrained: $\mathcal{S} = \{x \mid \ell \leq Ax \leq u\}$, where $A \in \mathbb{R}^{m \times n}$ and $\ell, u \in \mathbb{R}^m$.

Nonlinearly constrained: $\mathcal{S} = \{\, x \in \mathbb{R}^n \mid \ell \leq c(x) \leq u \,\}$, where $c : x \rightarrow c(x) \in \mathbb{R}^m$.

Common features of nonlinear programming algorithms

Iterative: produce a sequence of iterates $\{x_k\}$.

Greedy: for all $k \geq 0$, $f(x_{k+1}) \leq f(x_k)$.

Certification desired for nonlinear optimization/nonlinear programming algorithms:

- a guarantee that $\{x_k\}$ has a limit point that is a stationary point of the nonlinear optimization/nonlinear programming problem,
- a quantitative measure for the quality of the solution obtained upon termination of the search, and
- a rate of convergence.

Direct search algorithms for nonlinear programming:

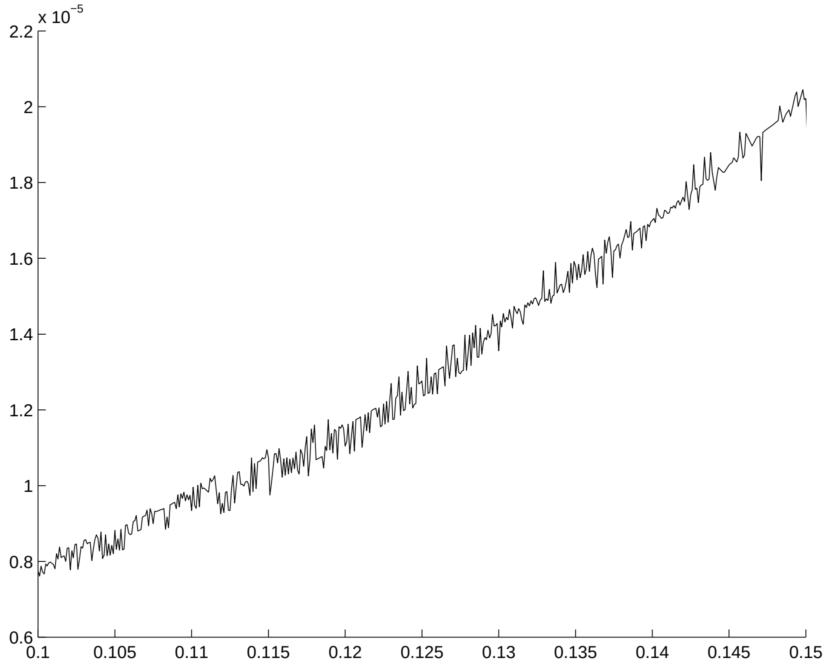
- Assume that while f is differentiable, ∇f and $\nabla^2 f$ are either unavailable or unreliable.

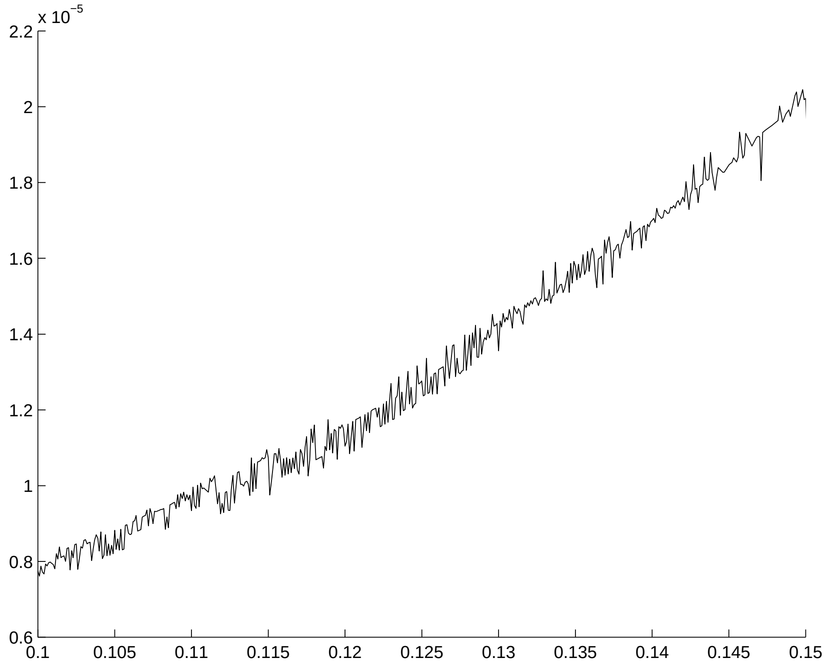
- Assume that the derivatives of the general equalities c are either unavailable or unreliable.

- Instead, use the values of $f(x)$ and $c(x)$ *directly* to drive the search.

Generating set search (GSS) methods are a special class of direct search algorithms that guarantee standard (first-order) convergence properties even in the absence of explicit derivative information.

An objective function afflicted with numerical noise deriving from an adaptive finite element scheme.





This example is a shape optimization problem for viscous channel flow.

Features of this problem:[1]

- The objective for this problem is to find a shape parameter to minimize the difference between straight channel flow and obstructed channel flow.

- The underlying infinite-dimensional problem is smooth.

- An adaptive finite element scheme is used to solve the stationary Navier–Stokes equations. Two adaptations were depicted.

- The oscillations diminish with successive adaptations, but the computed objective never becomes smooth, even near the minimizer.

[1]Sources: J. Borggard, D. Pelletier, and K. Vugrin, *On sensitivity analysis for problems with numerical noise*. AIAA Paper 2002–5553, 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.

J. Burkardt, M. Gunzburger, and J. Peterson, *Insensitive functionals, inconsistent gradients, spurious minima, and regularized functionals in flow optimization problems*, International Journal on Computational Fluid Dynamics, 16 (2002), pp. 171–185.

GSS methods for unconstrained optimization

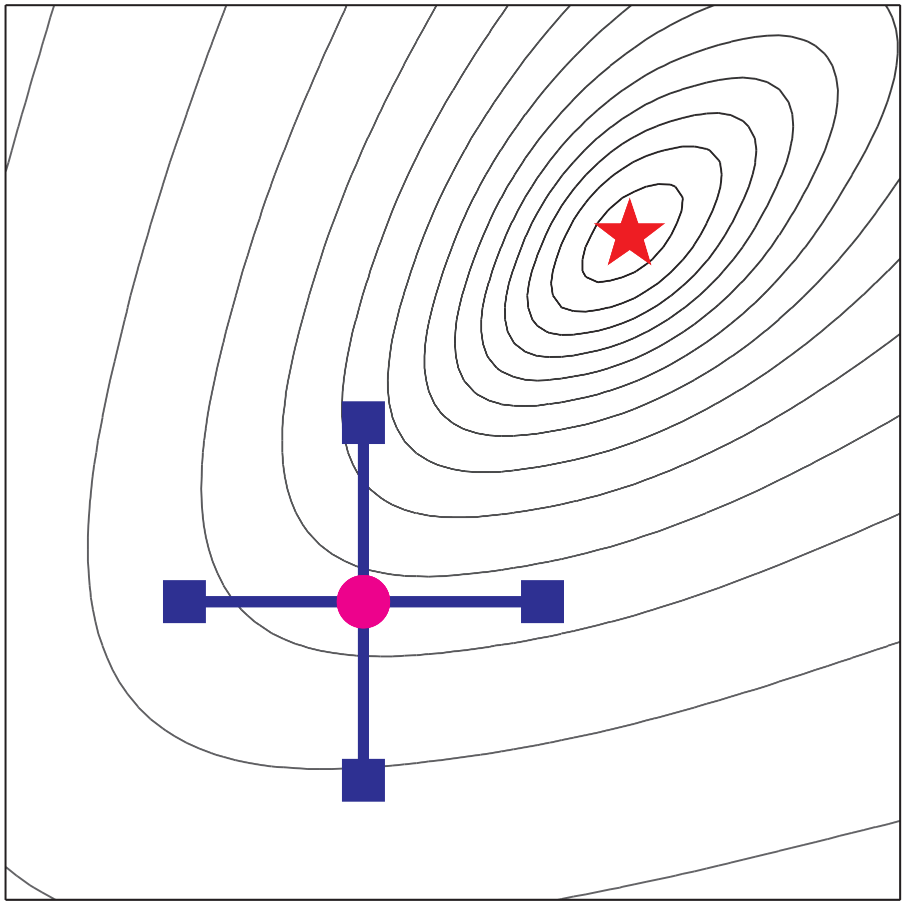
Look at one simple example applied to the modified Broyden tridiagonal function:

$$\underset{x \in \mathbb{R}^2}{\text{minimize}} \;\; f(x^1, x^2)$$

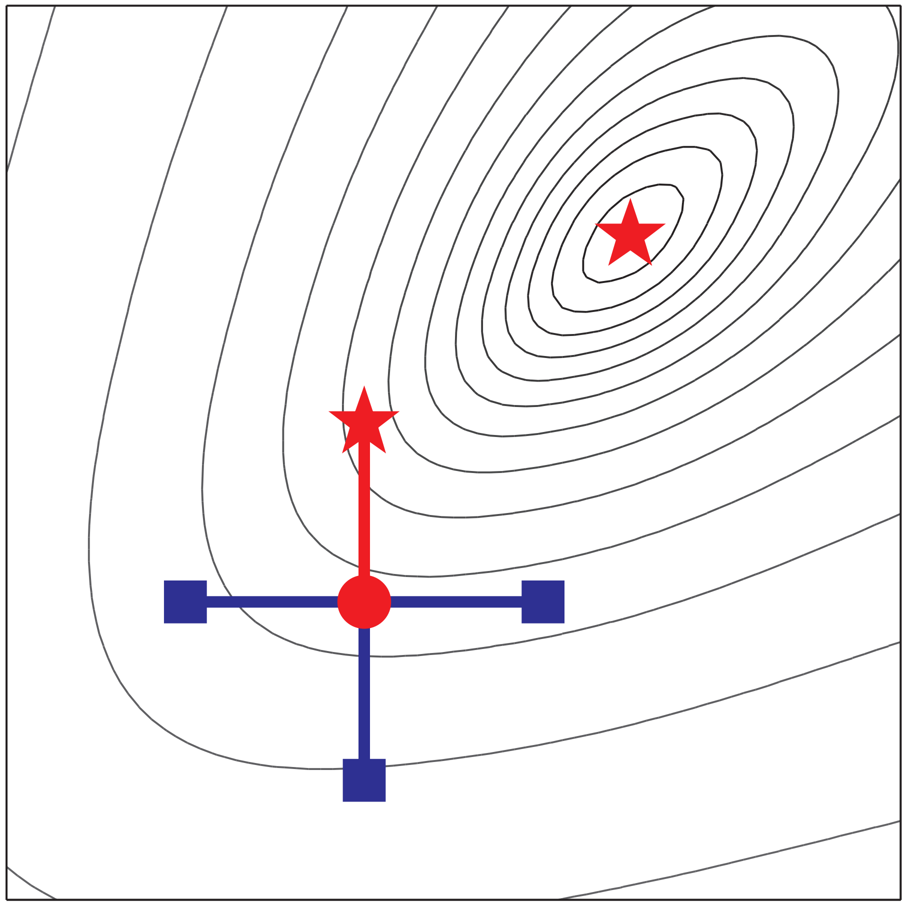
where

$$f(x) = \left|(3 - 2x^1)x^1 - 2x^2 + 1\right|^{\frac{7}{3}} + \left|(3 - 2x^2)x^2 - x^1 + 1\right|^{\frac{7}{3}}.$$

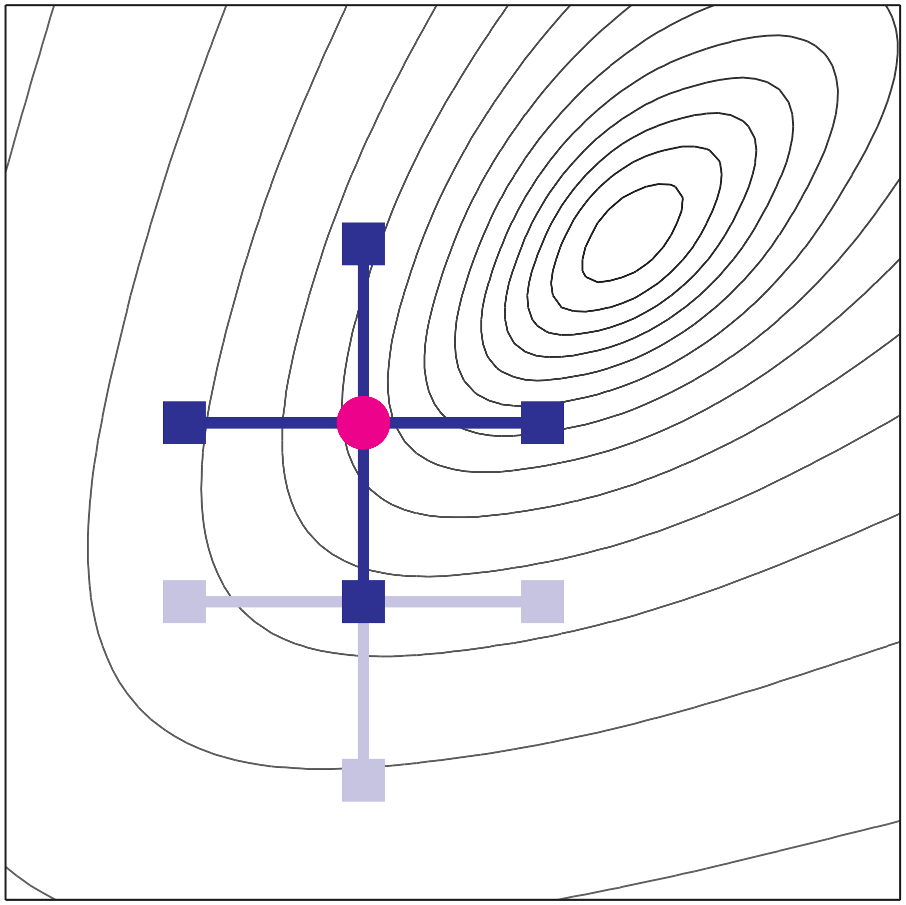
The initial configuration:



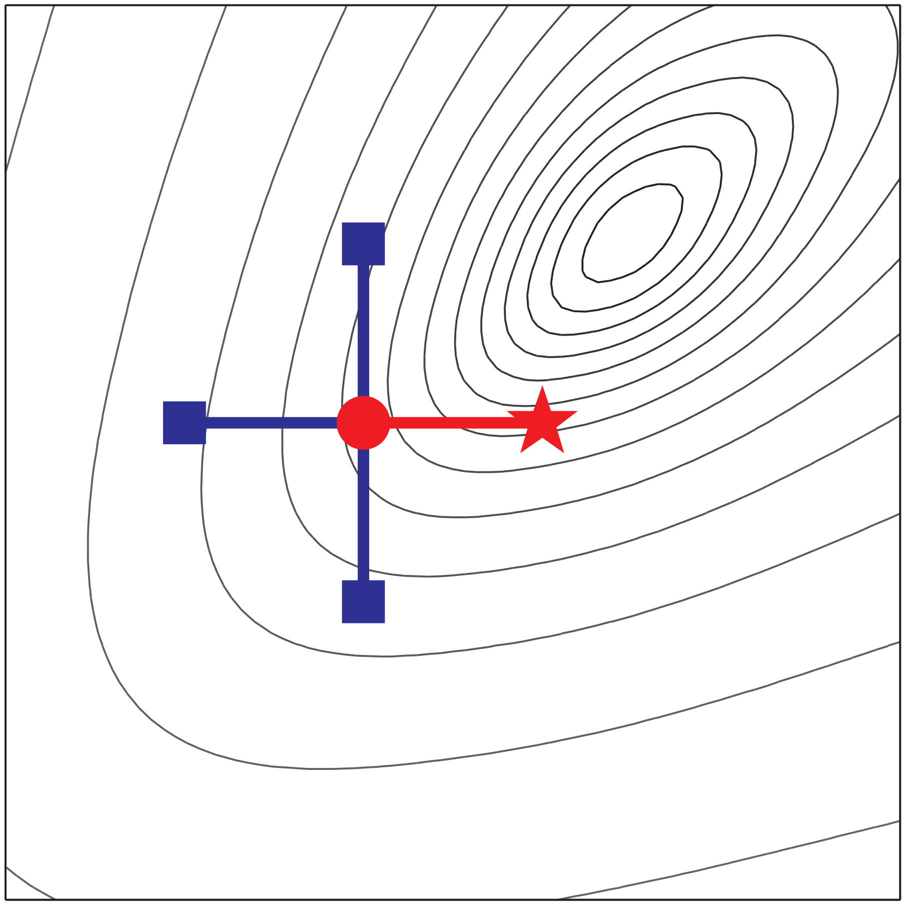
Identify improvement:



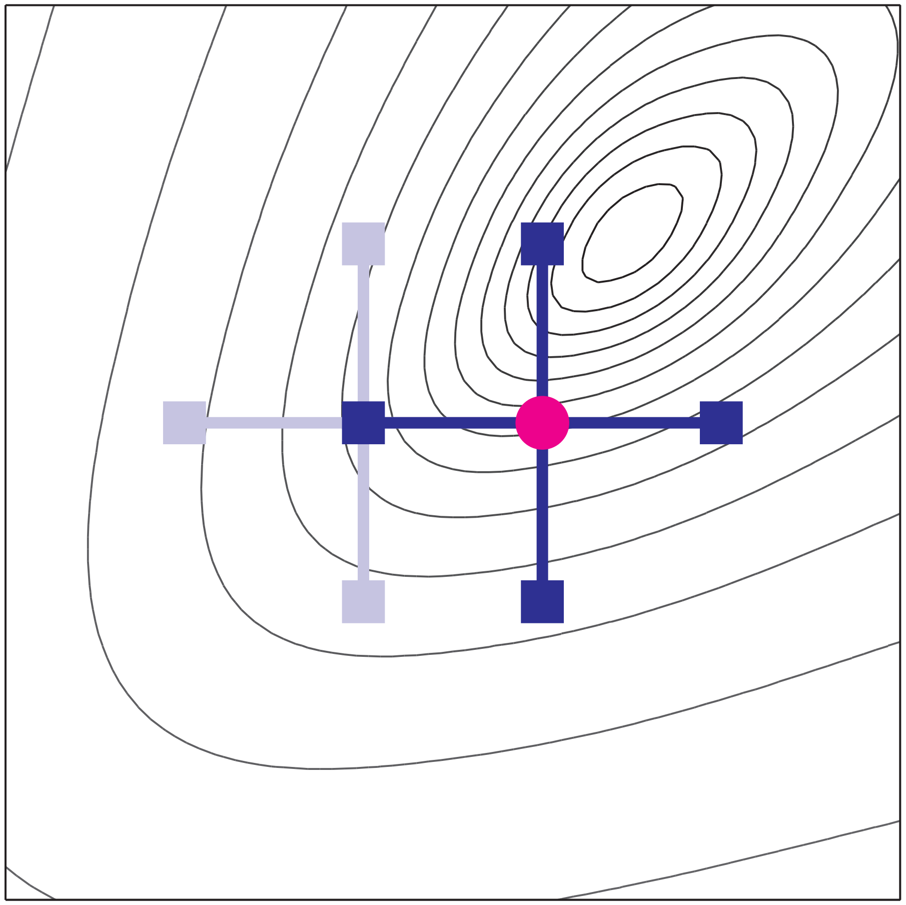
Move North; $k \in \mathcal{S}$



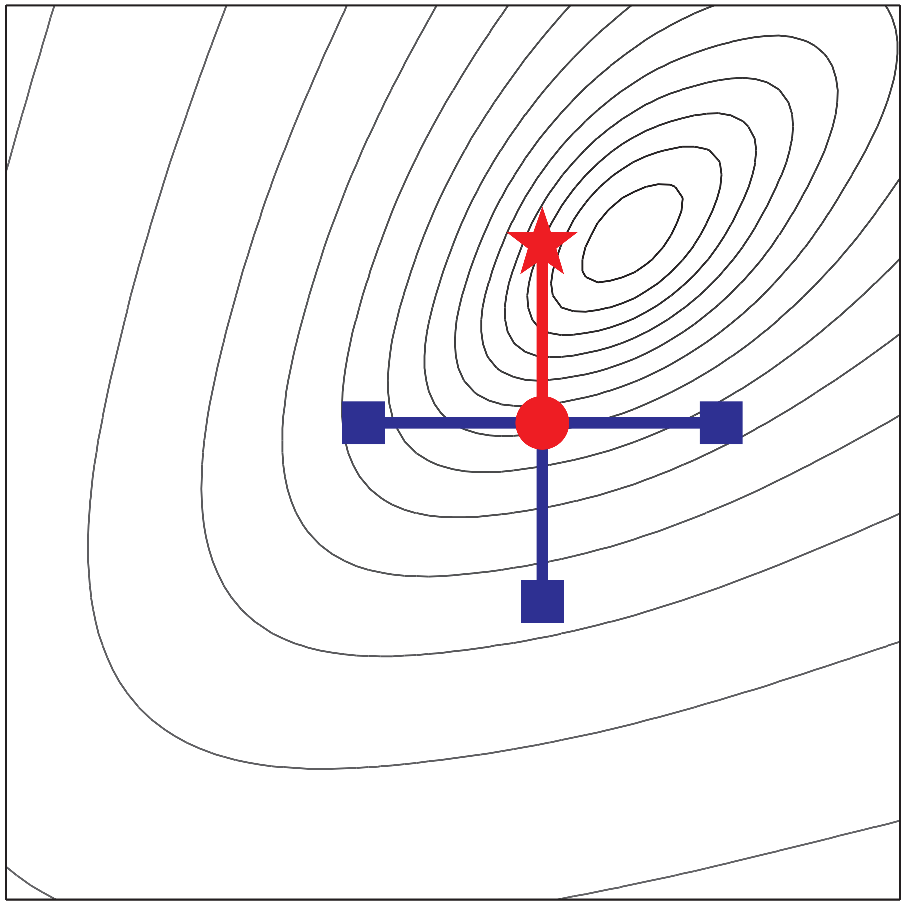
Identify improvement:



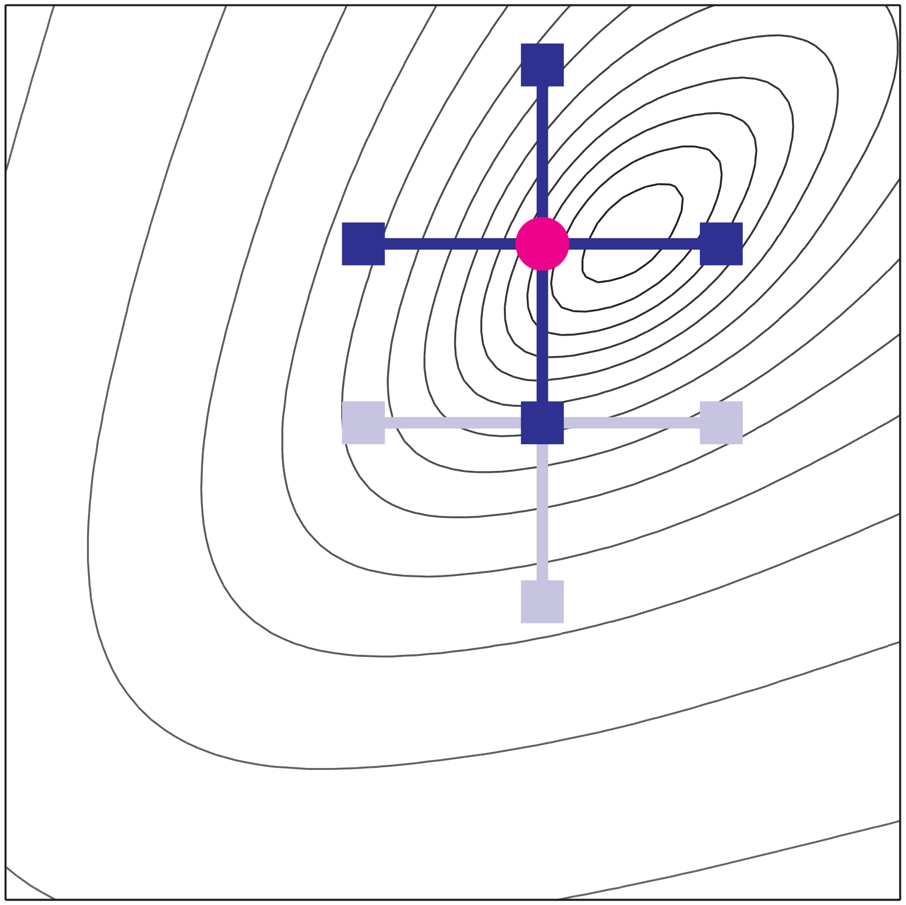
Move East; $k \in \mathcal{S}$



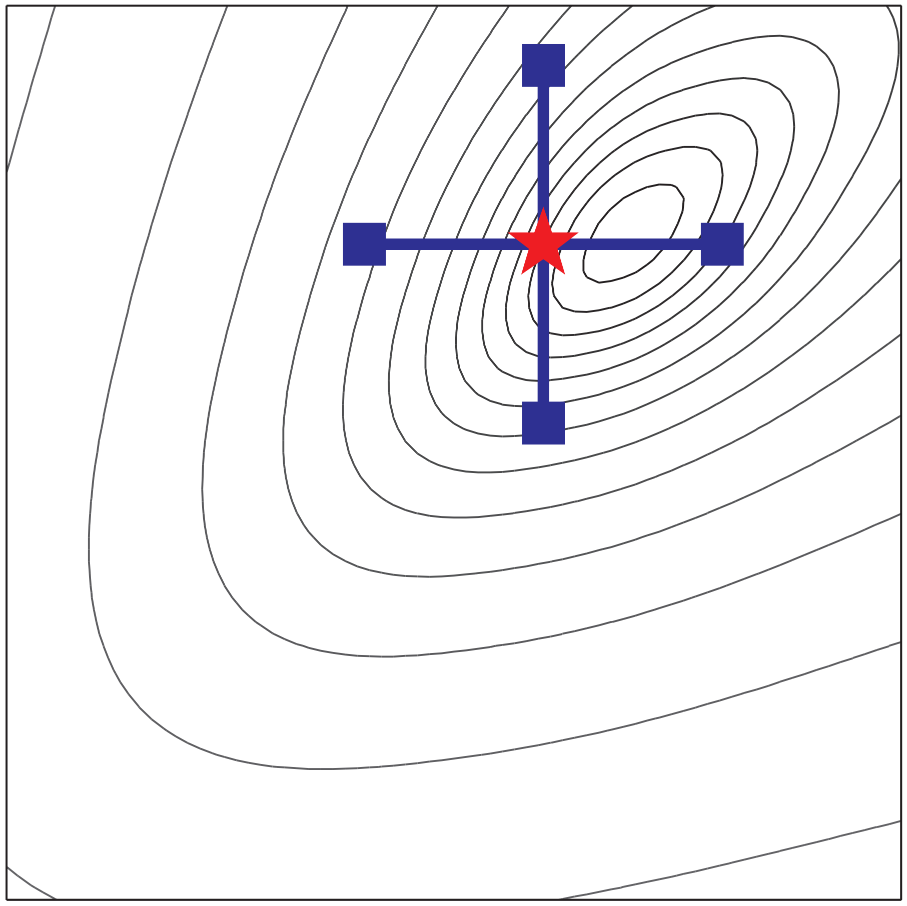
Identify improvement:



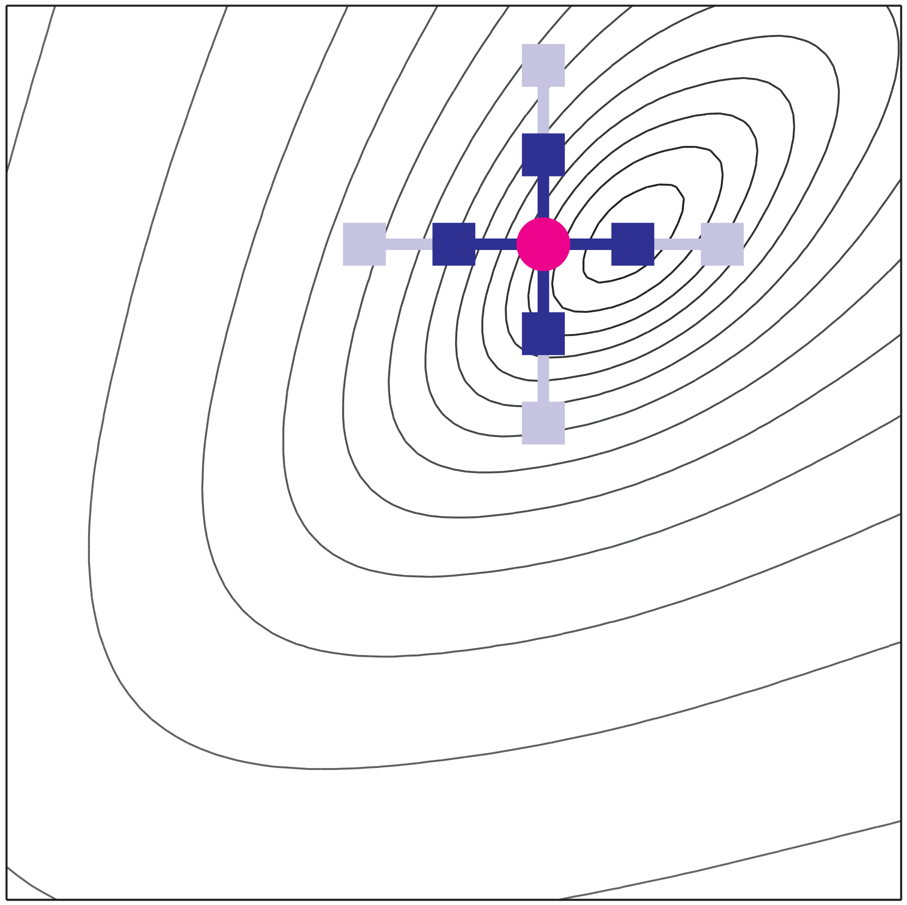
Move North; $k \in \mathcal{S}$



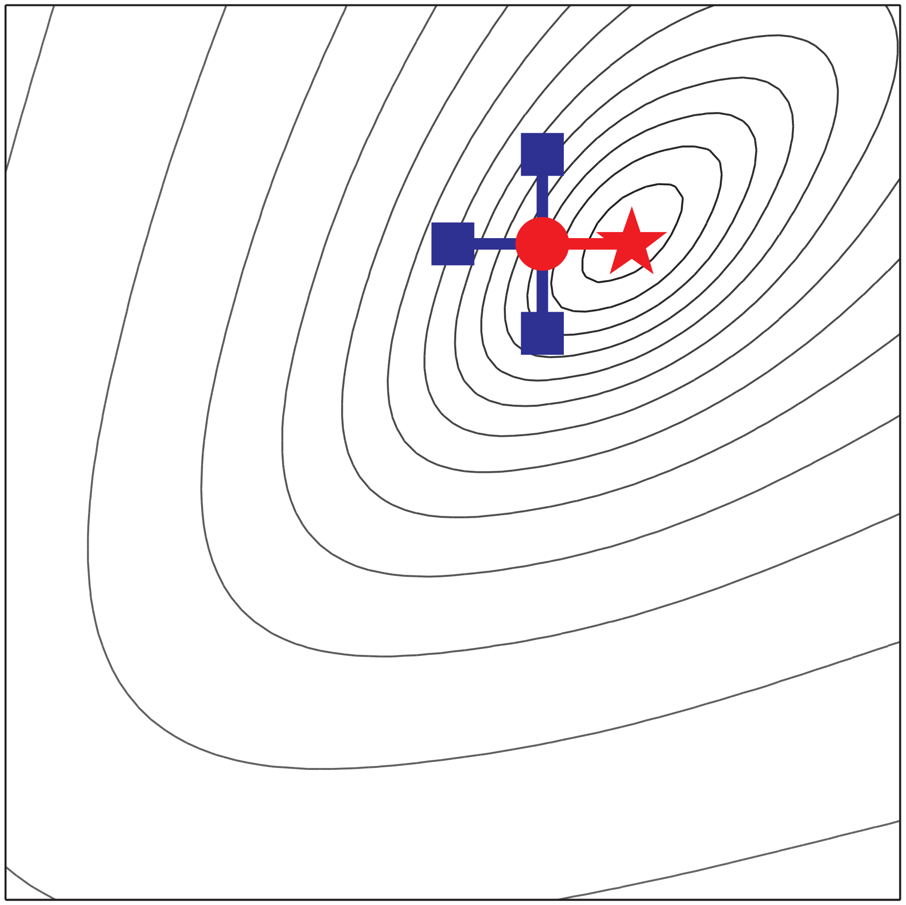
No improvement identified



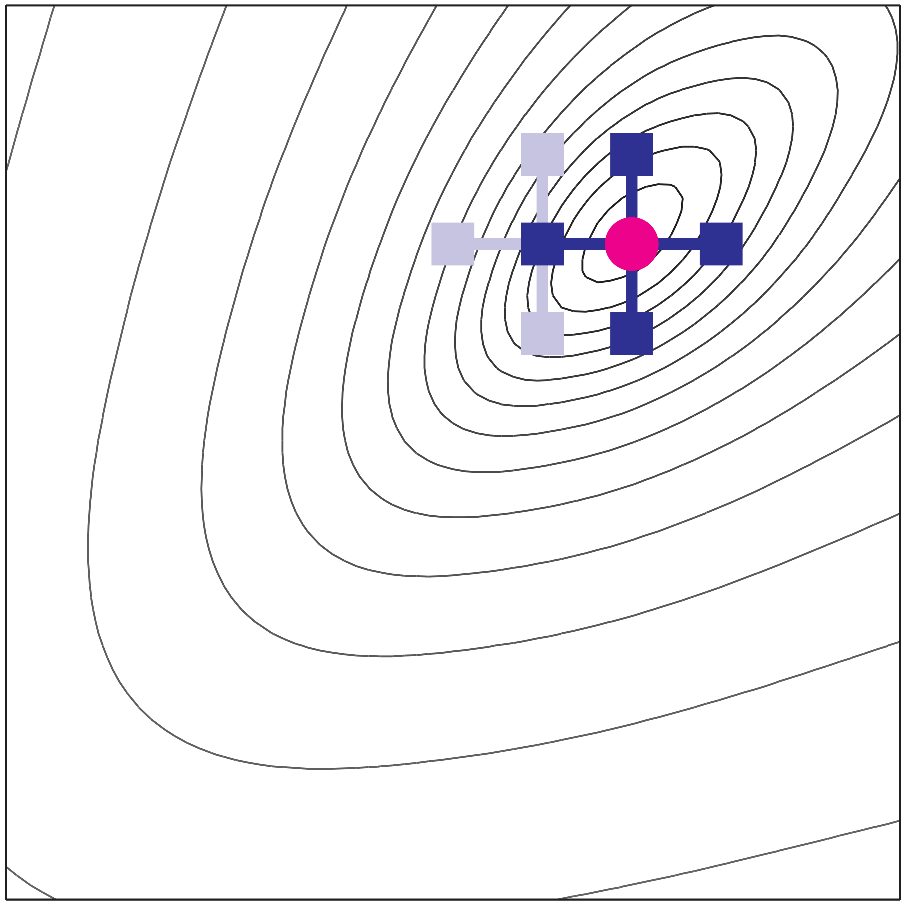
Reduce the lengths of the steps; $k \in \mathcal{U}$



Identify improvement



Move East; $k \in \mathcal{S}$....



First observation for the unconstrained case:

- At each iteration, a GSS method uses a *set* of search directions $\mathcal{D} = \{d^1, d^2, \ldots, d^r\}$.

- The set $\mathcal{D}$ forms a *positive spanning set* for $\mathbb{R}^n$: for any $y \in \mathbb{R}^n$

$$y = \alpha_1 d^1 + \cdots + \alpha_r d^r,$$

with $\alpha_i \geq 0$ for all $i \in \{1, \ldots, r\}$.

Second observation for the unconstrained case:

We can refine the requirement on the set of search directions $\mathcal{D}$ as follows:

$\mathcal{D}$ contains a subset $\mathcal{G} = \{g^1, g^2, \ldots, g^p\}$, $p \leq r$, where $\mathcal{G}$ forms a *positive basis* for for $\mathbb{R}^n$; i.e., for any $y \in \mathbb{R}^n$

$$y = \alpha_1 g^1 + \cdots + \alpha_r g^p,$$

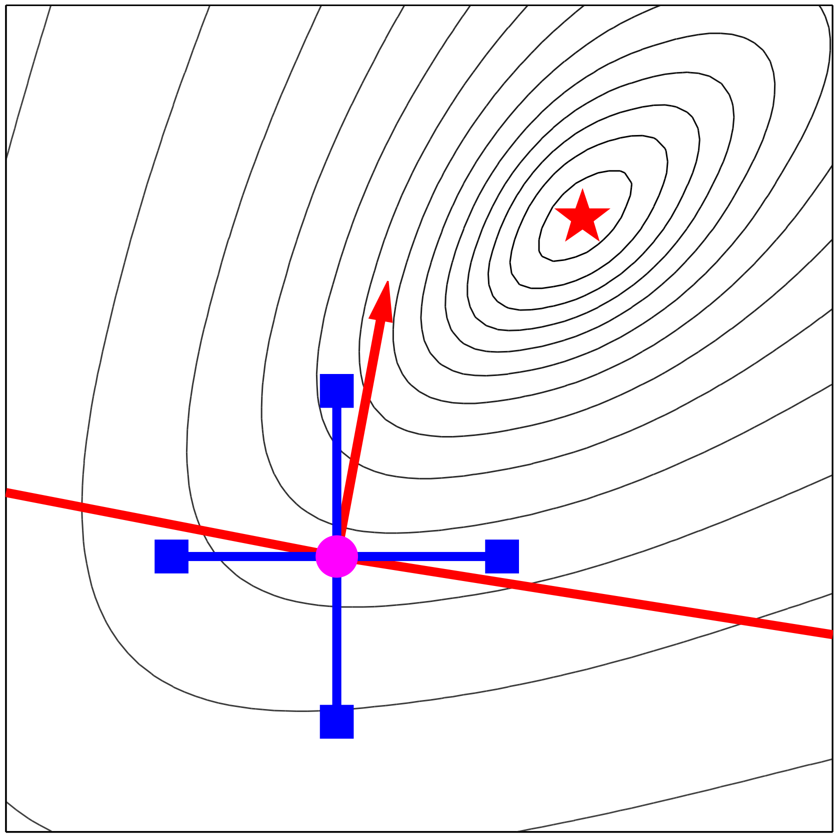
such that $\alpha_i \geq 0$, for all $i \in \{1, \ldots, p\}$ and

$$g^i \neq \sum_{j \in \{1,\ldots,p\} \setminus \{i\}} \alpha'_j g^j.$$

.

Fact: $n + 1 \leq |\mathcal{G}| \leq 2n$.

The inclusion of a positive basis means that GSS methods are gradient-related:



Minimal requirements for GSS methods for unconstrained minimization:

- At each iteration the set of search directions $\mathcal{D}_k$ includes a positive basis $\mathcal{G}_k$ for $\mathbb{R}^n$.
- The step-length control parameter Δ_k is reduced only when no descent is identified for the step of length Δ_k along the directions $g_k^i \in \mathcal{G}_k$; i.e., $f(x_k) \leq f(x_k + \Delta_k g_k^i)$ for all $g_k^i \in \mathcal{G}_k$ (i.e., when $k \in \mathcal{U}$).

These are the requirements that make certification possible.

Certification for GSS methods for unconstrained optimization:

- $\liminf_{k \to \infty,\ k \in \mathcal{U}} \| \nabla f(x_k) \| = 0$.
- $\| \nabla f(x_k) \| = O(\Delta_k)$ for $k \in \mathcal{U}$.
- Rate of convergence is linear for $\{ x_k \}_{k \in \mathcal{U}}$.

Flexibility in devising GSS methods for unconstrained minimization:

- At each iteration the set of search directions $\mathcal{D}_k$ may include directions in additional to a positive basis $\mathcal{G}_k$ of $\mathbb{R}^n$; e.g., $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$.
- For any $d_k^i \in \mathcal{D}_k$, it is possible to consider steps of the form $c_k^i \, \Delta_k \, d_k^i$.
- So long as $f(x_k + c_k^* \, \Delta_k \, d_k^*) < f(x_k) + \rho(\Delta_k)$ (i.e., sufficient improvement on $f(x_k)$ is found), either $d_k^* \in \mathcal{G}_k$ or $d_k^* \in \mathcal{H}_k$ is acceptable.
- The direction d_k^* need not be a *descent direction*.

This is the flexibility that allow heuristics both for acceleration schemes and for global optimization.

Fundamental strategy for incorporating heuristics:

At each iteration, divide the search into two phases:

Phase 1. Undertake exploration based on the search directions in $\mathcal{H}_k$.

This is the phase that supports the incorporation of heuristics.

Phase 2. Poll the steps defined by the search directions in $\mathcal{G}_k$.

This is the phase that ensures the convergence results hold.

Phase 1:

- Choose $\mathcal{H}_k$.
- Evaluate $f(x_k + c_k^i \, \Delta_k \, h_k^i)$ for some finite number of $h_k^i \in \mathcal{H}_k$.
- If an h_k^i for which

$$f(x_k + c_k^i \, \Delta_k \, h_k^i) < f(x_k) + \rho(\Delta_k)$$

  is found, it is possible to set $x_{k+1} = x_k + c_k^i \, \Delta_k \, h_k^i$, $k \in \mathcal{S}$, and $k = k+1$ (i.e., skip **Phase 2**).
- Else, go to **Phase 2**.

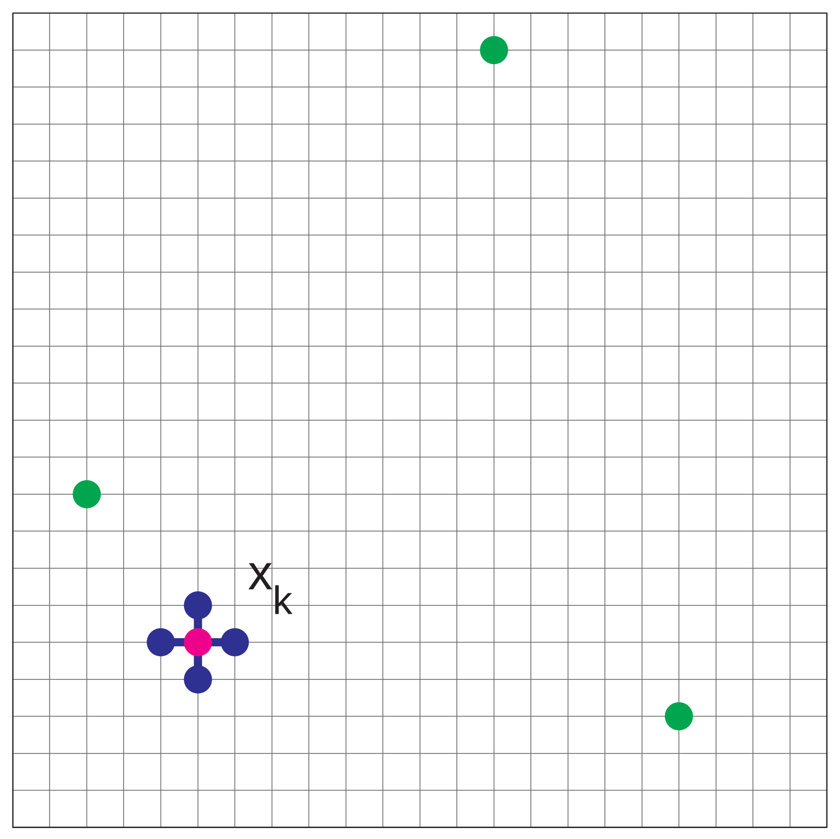
Phase 2:

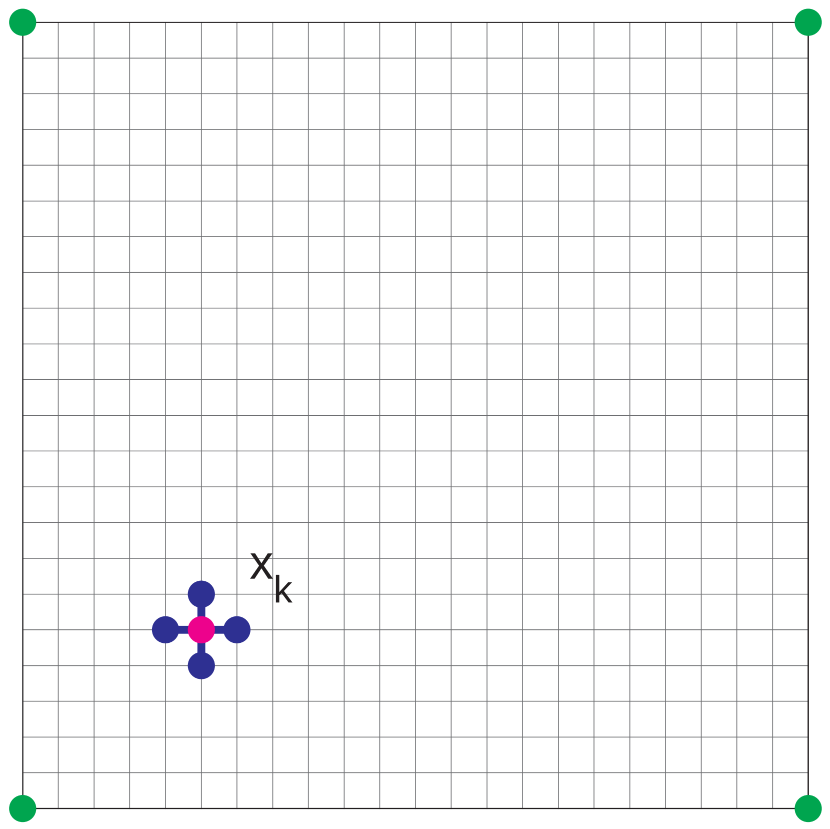
- Choose $\mathcal{G}_k$.
- Evaluate $f(x_k + c_k^i \, \Delta_k \, g_k^i)$ until either find a g_k^i for which

$$f(x_k + c_k^i \, \Delta_k \, g_k^i) < f(x_k) + \rho(\Delta_k) \tag{1}$$

  or determine that there no such $g_k^i \in \mathcal{G}_k$ satisfying (1).
- If find at least one $g_k^i \in \mathcal{G}_k$ satisfying (1), then set $x_{k+1} = x_k + c_k^i \, \Delta_k \, g_k^i$ and $k \in \mathcal{S}$.
- Else set $x_{k+1} = x_k$, $k \in \mathcal{U}$, and $\Delta_{k+1} = \theta \Delta_k$, $\theta \in (0,1)$.
- Set $k = k + 1$.

Examples of points to consider in Phase 1





Two examples in $\mathbb{R}^2$ of using oracles to try to predict successful steps.

GSS methods for bound constrained optimization

Again look at one simple example applied to the modified Broyden tridiagonal function:

$$\underset{x \in \mathbb{R}^2}{\text{minimize}} \; f(x^1, x^2)$$

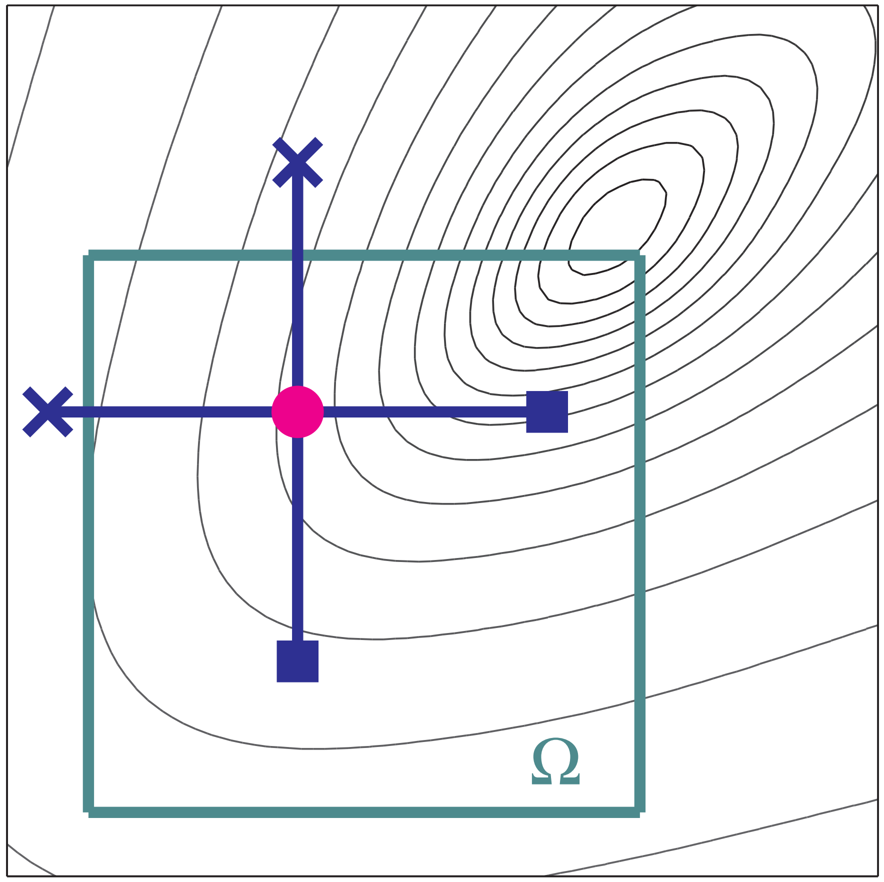
where

$$f(x) = \left|(3 - 2x^1)x^1 - 2x^2 + 1\right|^{\frac{7}{3}} + \left|(3 - 2x^2)x^2 - x^1 + 1\right|^{\frac{7}{3}},$$

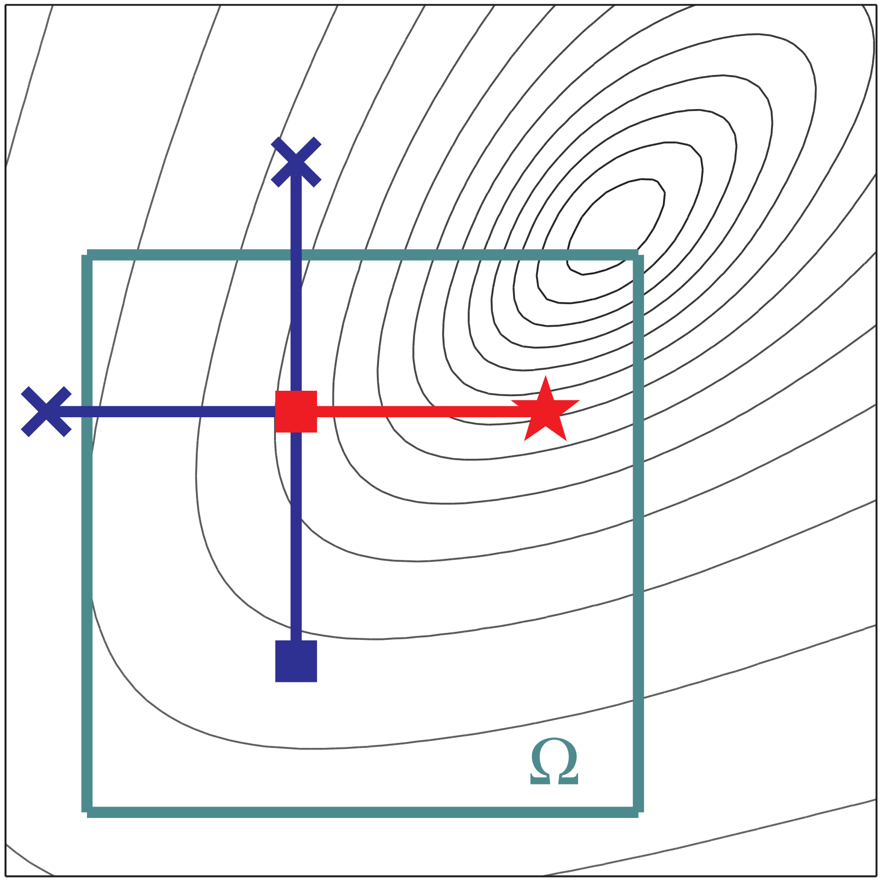
But now add bound ("box") constraints.

Use a *feasible iterates* approach.

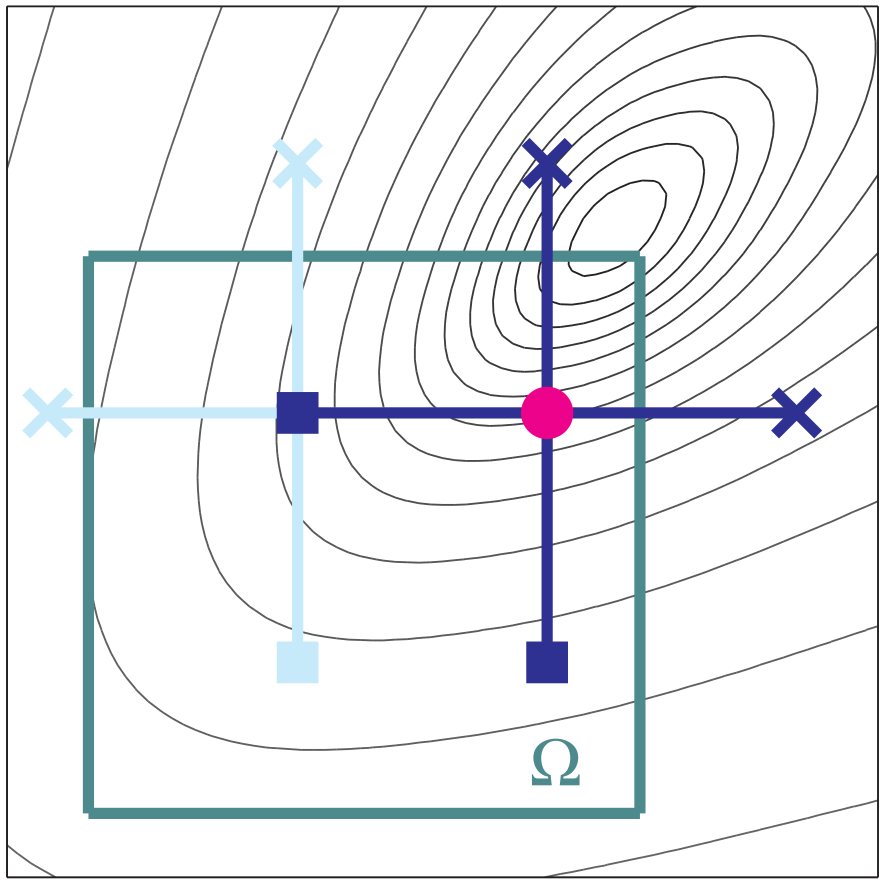
The initial configuration:



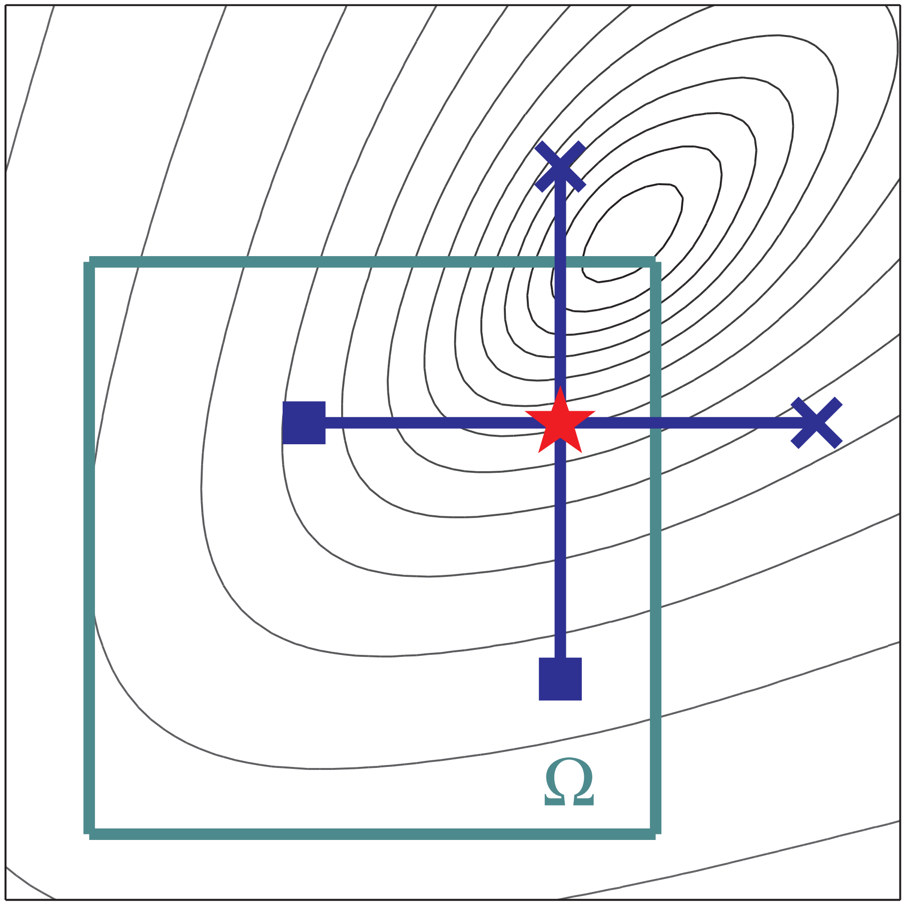
Identify feasible improvement



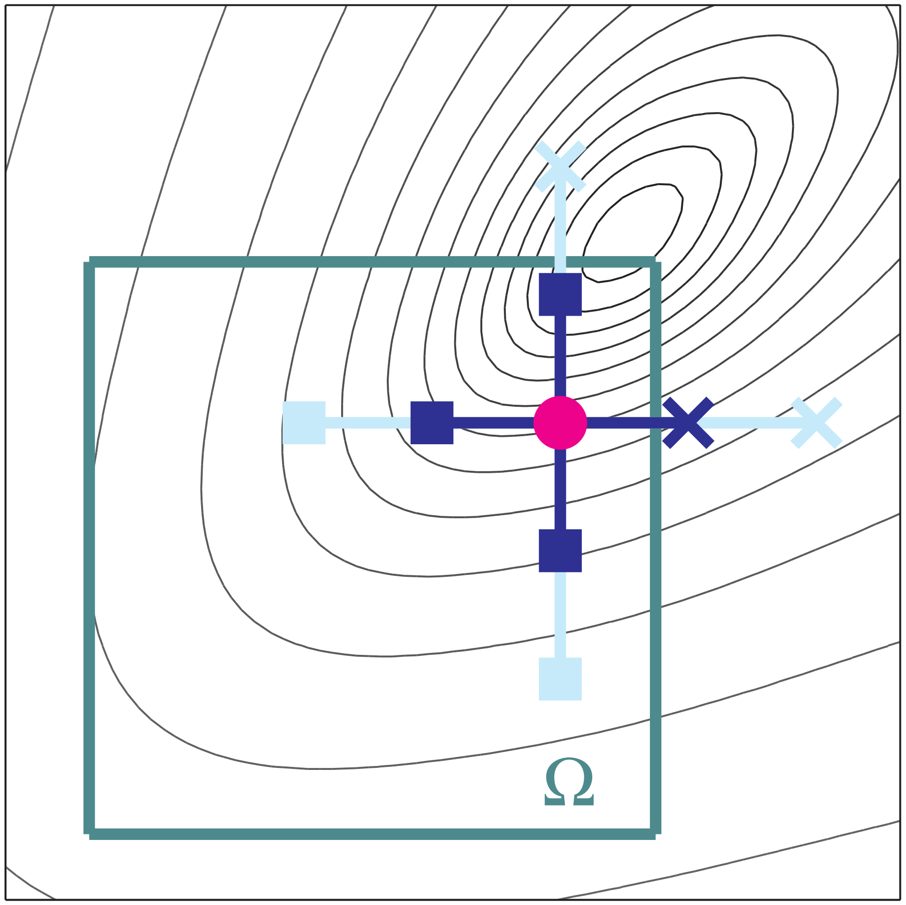
Move East; $k \in \mathcal{S}$



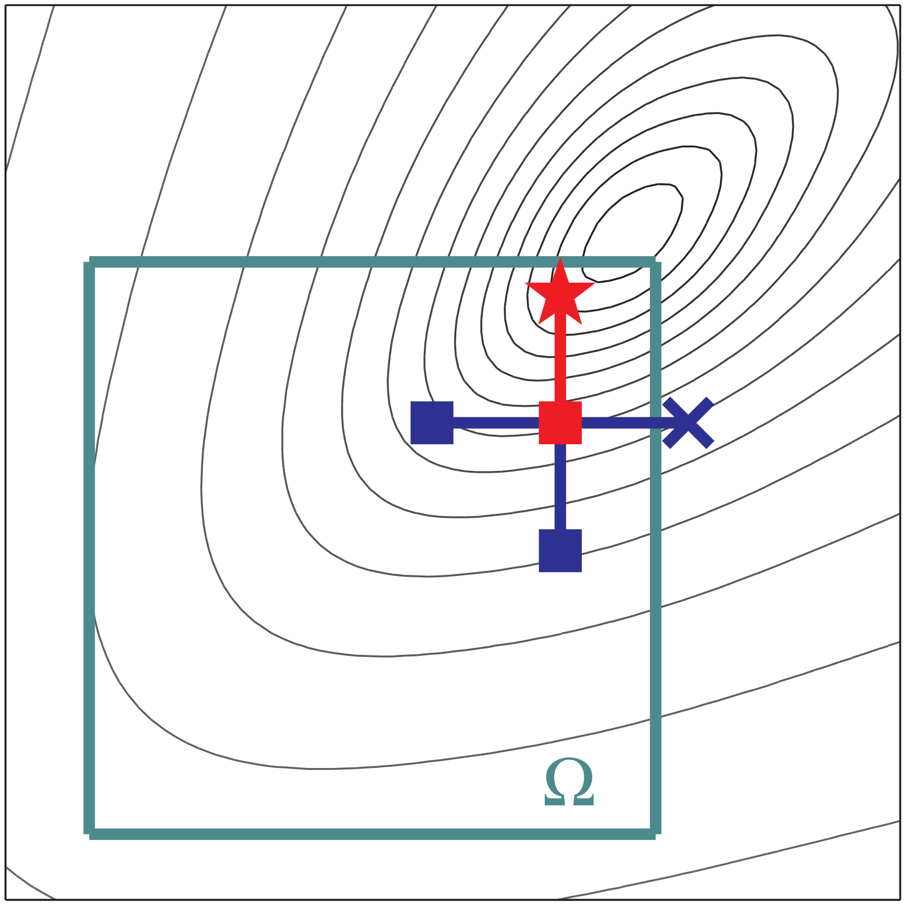
No feasible improvement identified



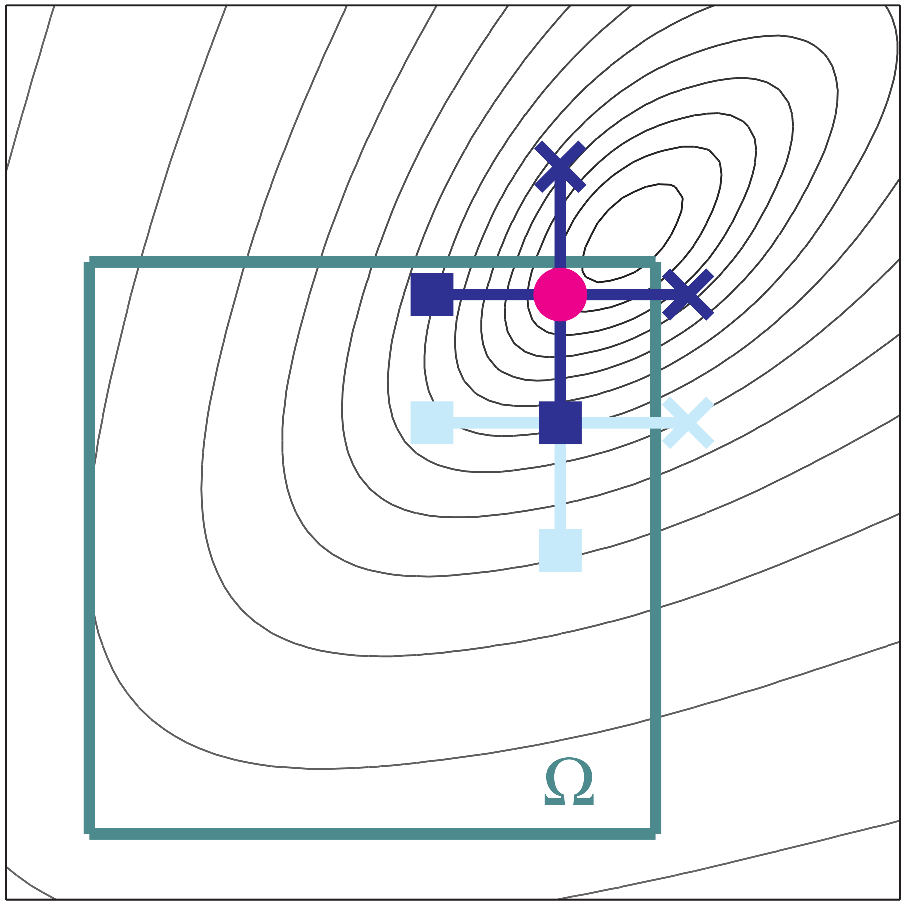
Contract; $k \in \mathcal{U}$



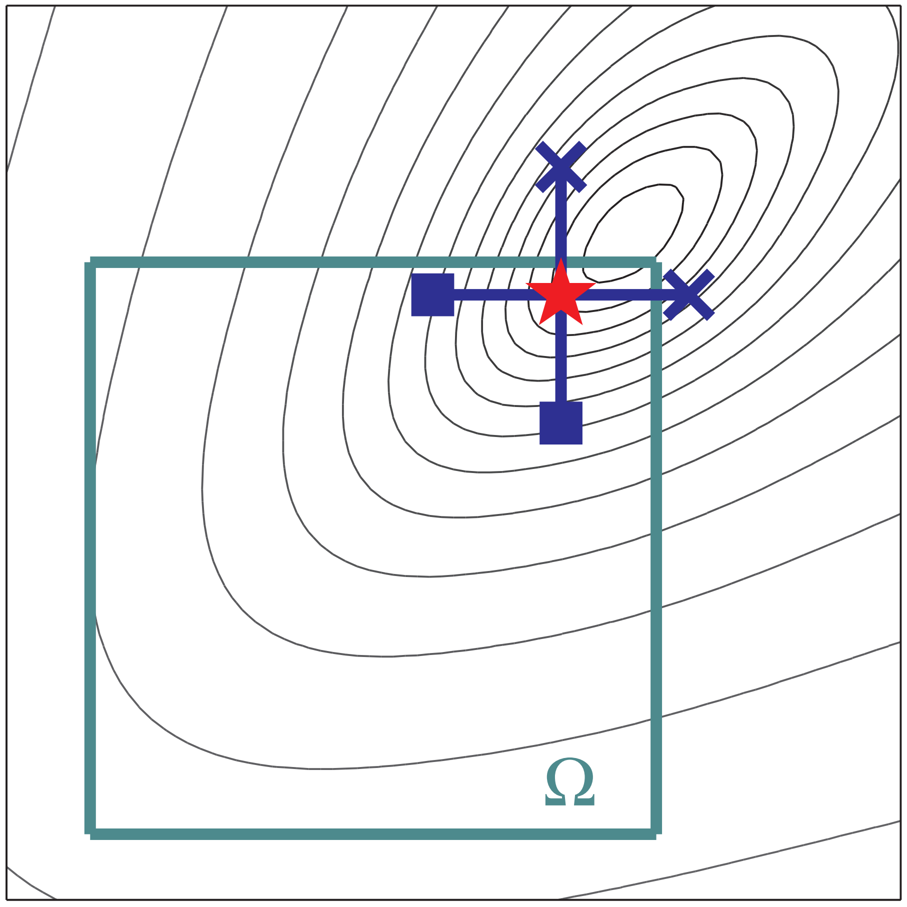
Identify feasible improvement



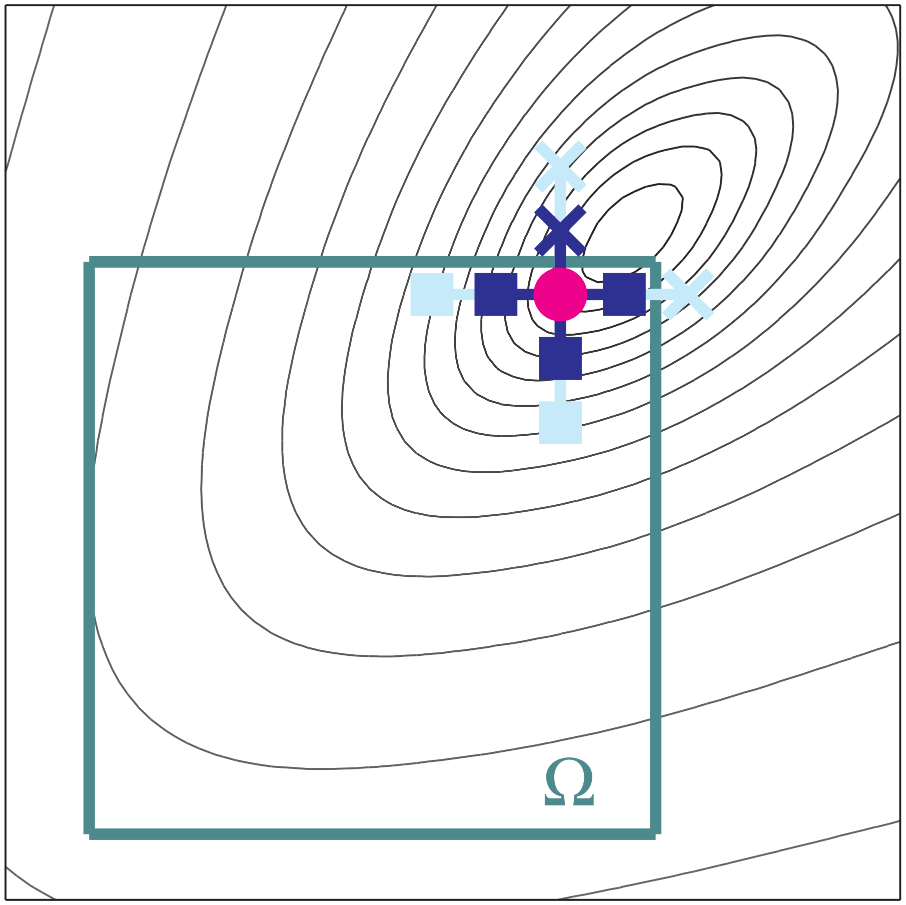
Move North; $k \in \mathcal{S}$



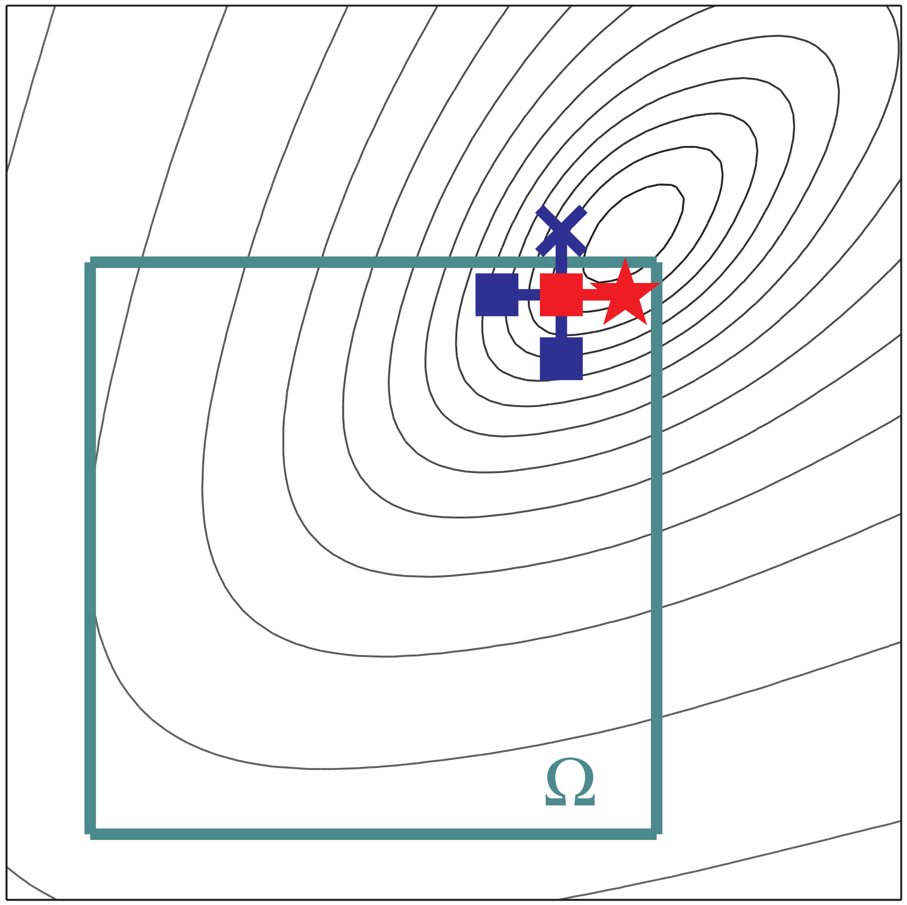
No feasible improvement identified



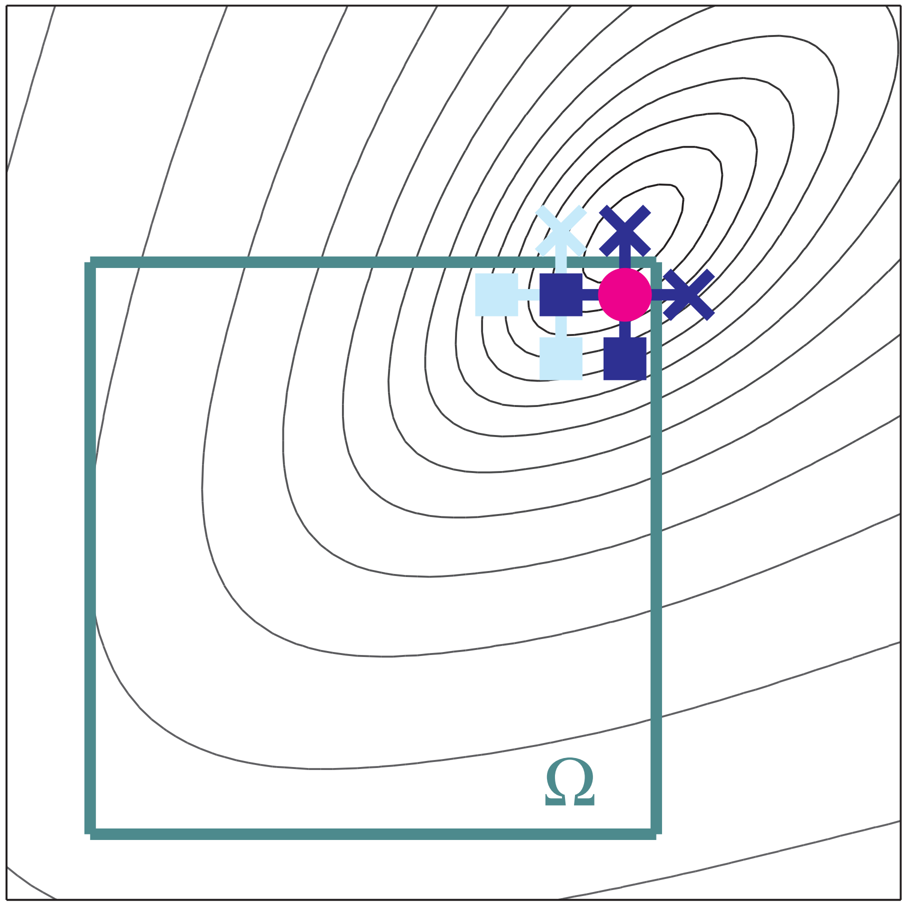
Contract; $k \in \mathcal{U}$



Identify feasible improvement



Move East; $k \in \mathcal{S}$....



“Minimal” requirements for GSS methods for bound constrained minimization:

- the set of search directions $\mathcal{D}_k$ includes the set $\mathcal{G} = \{\pm e^i \,|\, i = 1, \ldots, n\}$.
- the step-length control parameter Δ_k is reduced only when no feasible descent is identified for the step of length Δ_k along the directions $g^i \in \mathcal{G}$ (i.e., when $k \in \mathcal{U}$).

Certification for GSS methods for bound constrained optimization is equivalent to that for the unconstrained case, with an appropriately chosen measure for bound-constrained stationarity.

Flexibility in devising GSS methods for bound constrained minimization:

- At each iteration the set of search directions $\mathcal{D}_k$ may include directions in addition to the set $\mathcal{G} = \{\pm e^i \,|\, i = 1, \ldots, n\}$; e.g., $\mathcal{D}_k = \mathcal{G} \cup \mathcal{H}_k$.

- For any $d_k^i \in \mathcal{D}_k$, it is possible to consider steps of the form $c_k^i \, \Delta_k \, d_k^i$.

- So long as $f(x_k + c_k^* \, \Delta_k \, d_k^*) < f(x_k) + \rho(\Delta_k)$ (i.e., sufficient improvement on $f(x_k)$ is found) and $(x_k + c_k^* \, \Delta_k \, d_k^*) \in \Omega$ (i.e., the step is feasible), either $d_k^* \in \mathcal{G}$ or $d_k^* \in \mathcal{H}_k$ is acceptable.

- The direction d_k^* need not be a *feasible descent direction*.

This admits heuristics for both acceleration schemes and global optimization.

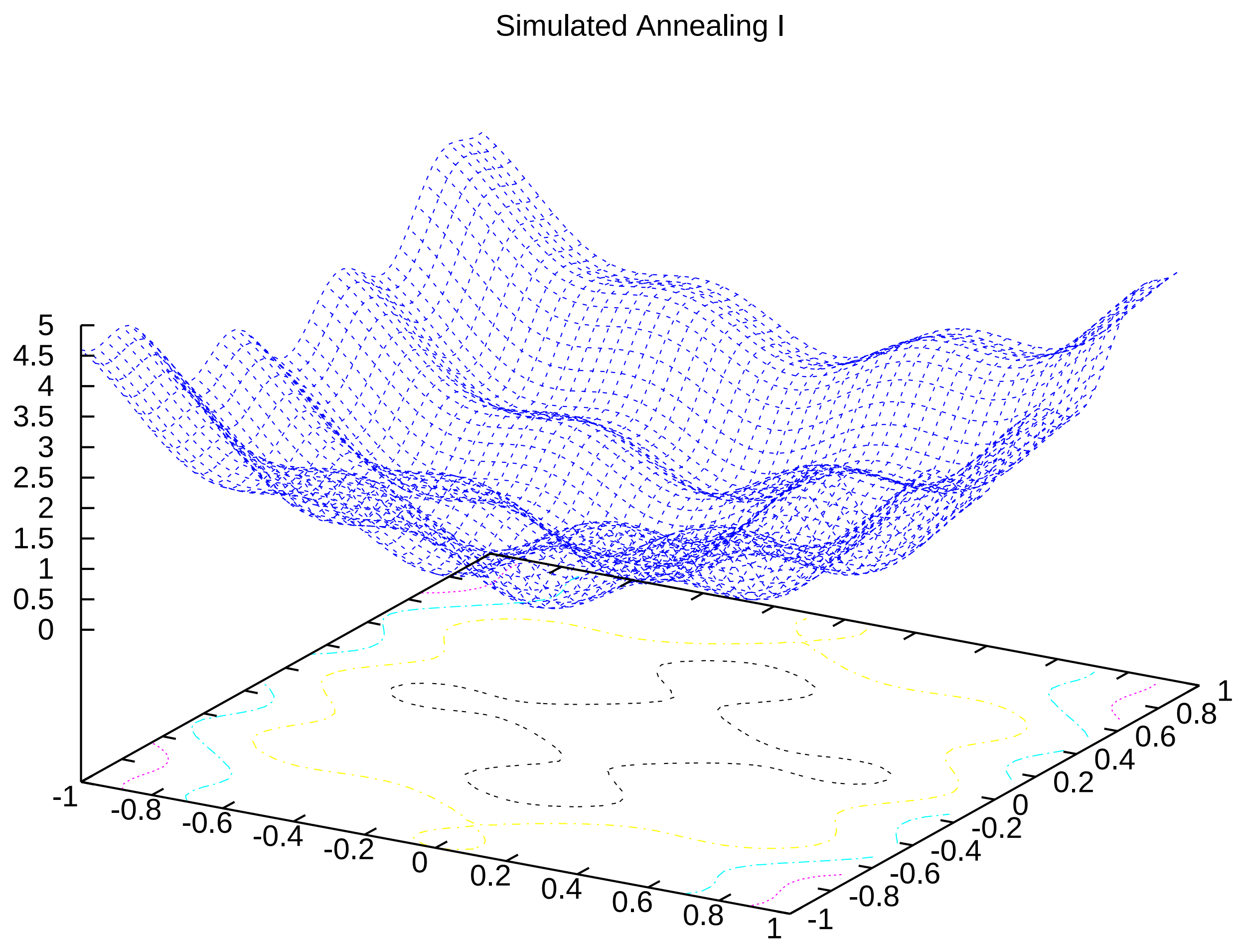
One strategy for trying to "globalize" the search:

- Use approximations $\hat{f}_k$ of the objective f to accelerate generating set search.

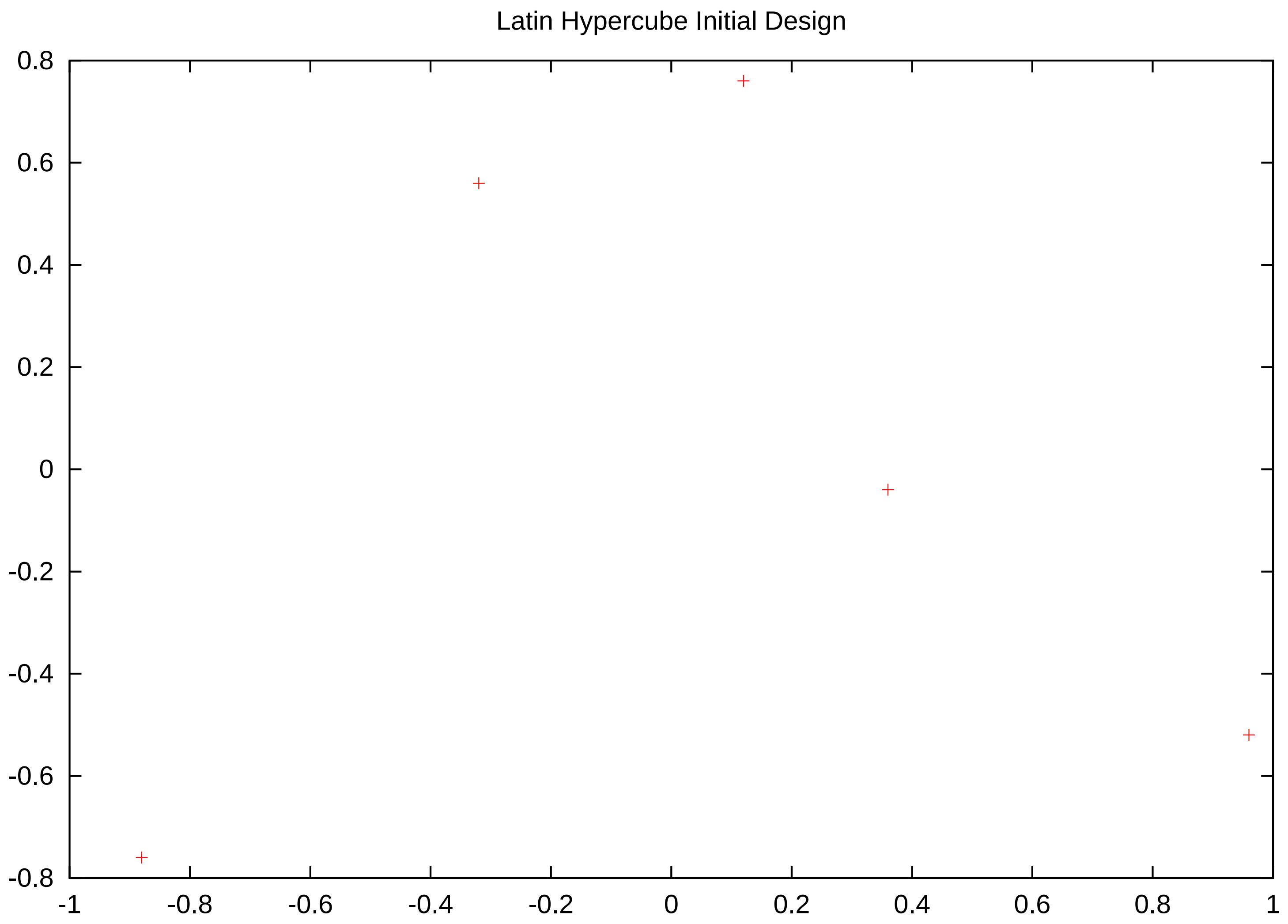
- Use search criteria S_k to encourage a wider exploration of the feasible region.

- Choose both $\hat{f}_k$ and S_k in a way that preserves the theoretical guarantees of generating set search provided by the large body of analysis and takes advantage of our growing computational experience.

The objective f graphed over the feasible region

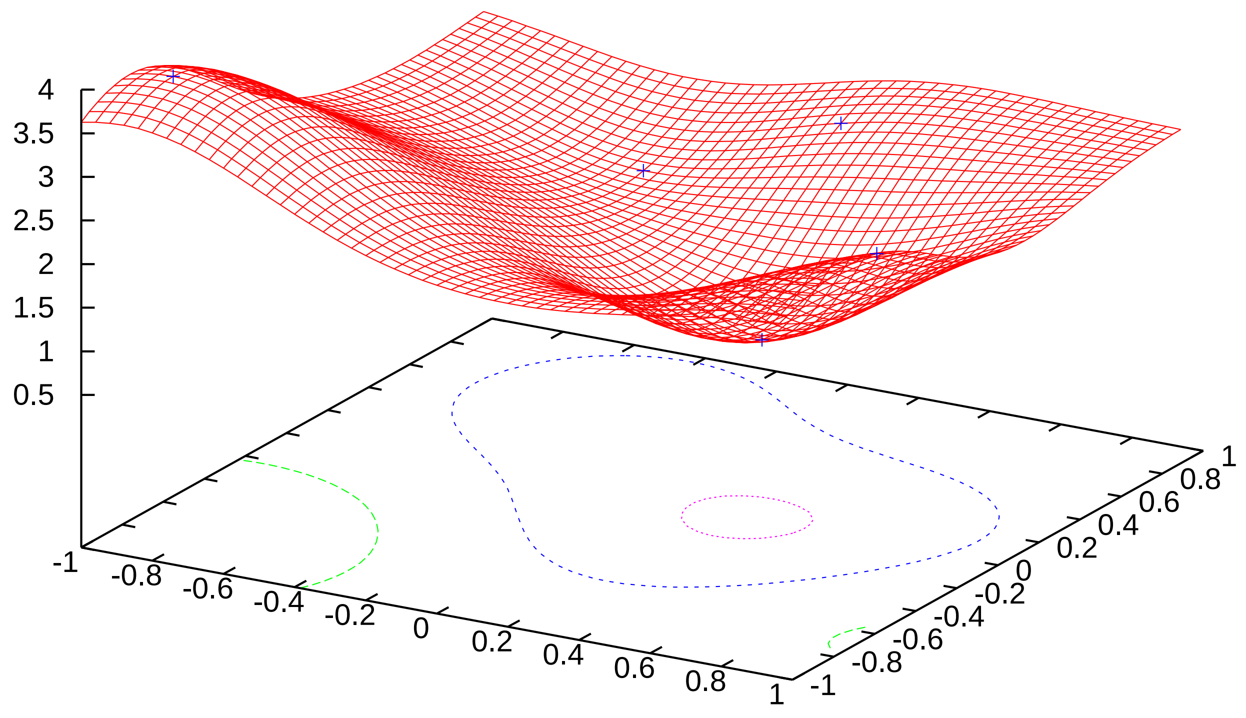


The initial design sites $x^1, \ldots, x^5$, selected from the feasible region

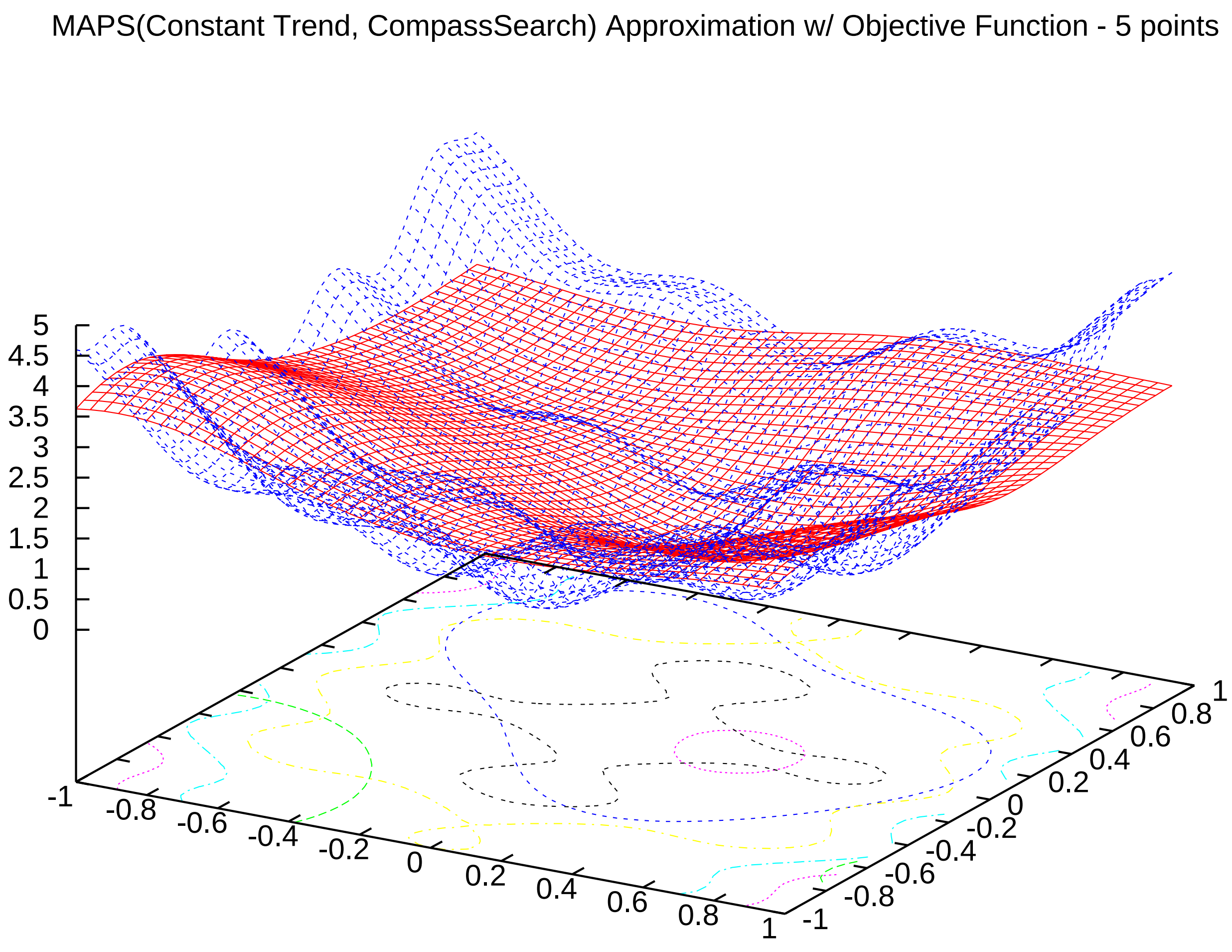


The initial approximation $\hat{f}_0$ graphed over the feasible region

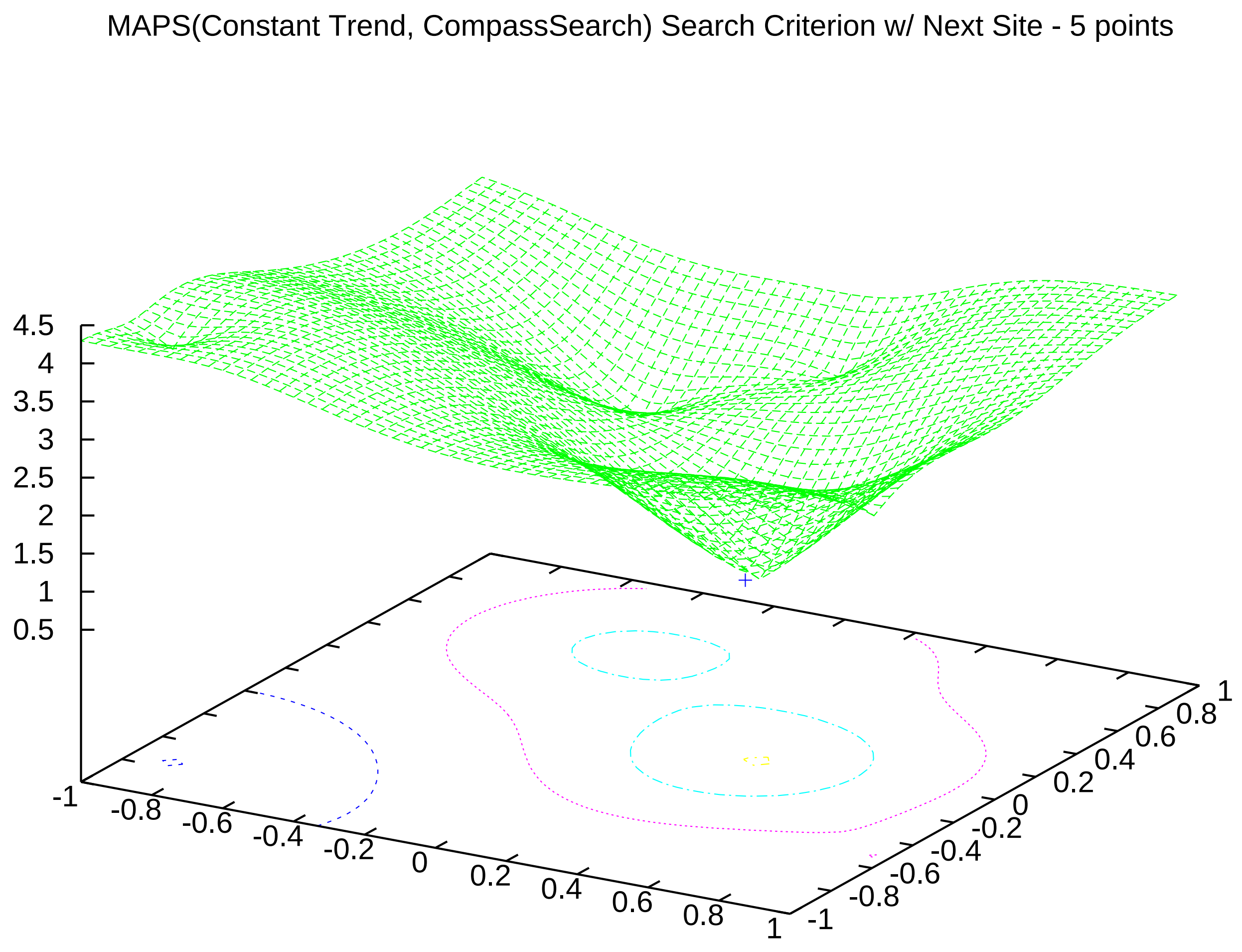
MAPS(Constant Trend, CompassSearch) Approximation - 5 points



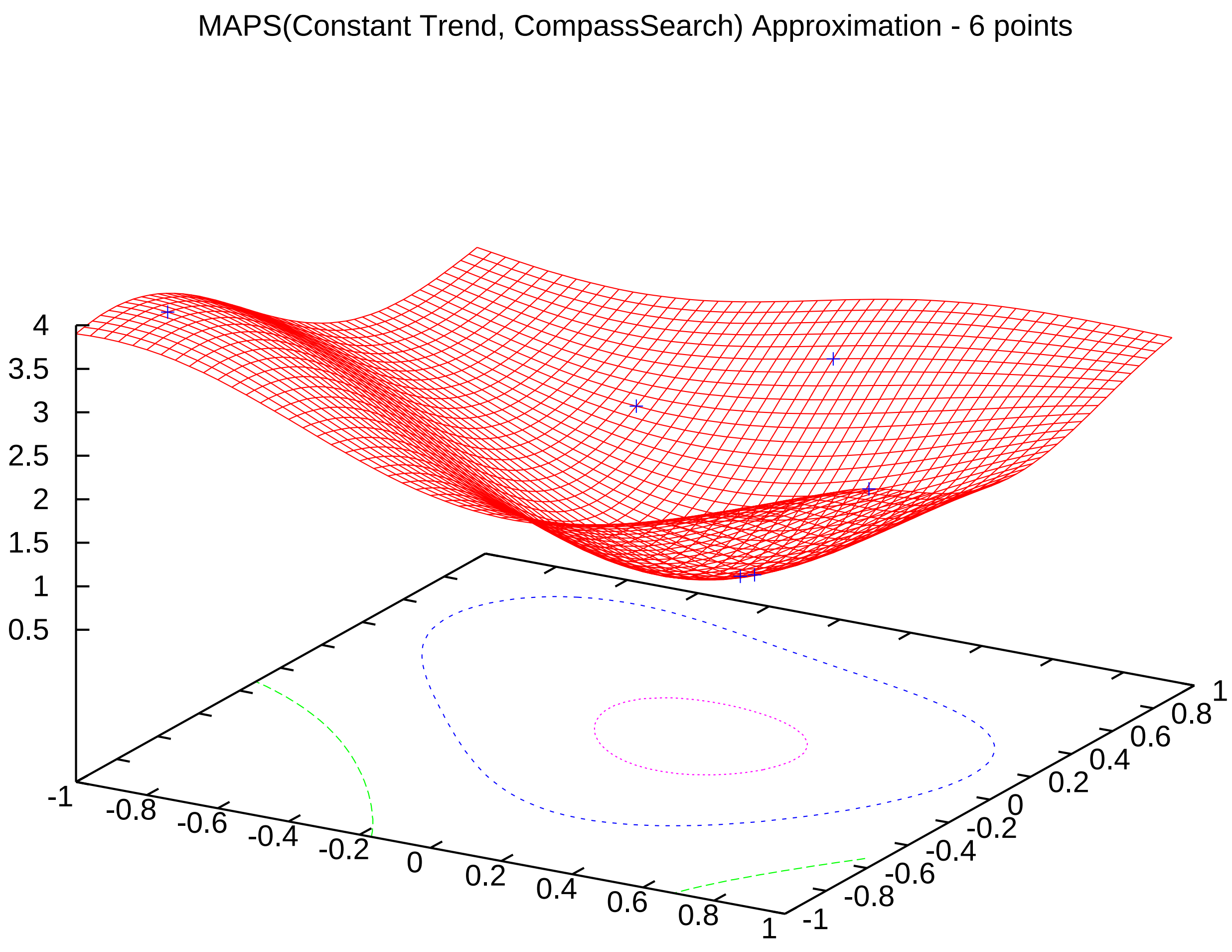
The objective f and the initial approximation $\hat{f}_0$



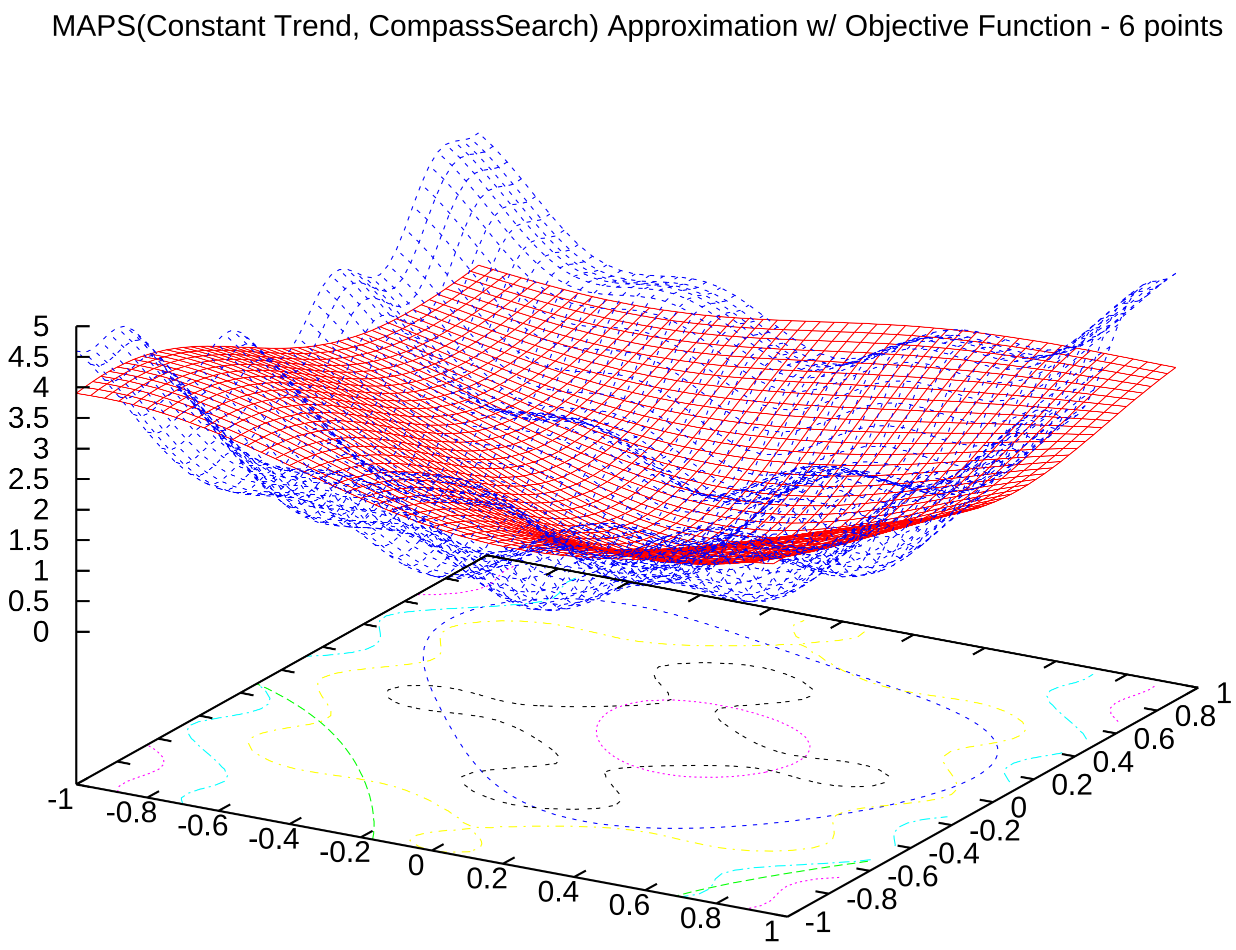
The initial search criterion S_0 graphed over the feasible region



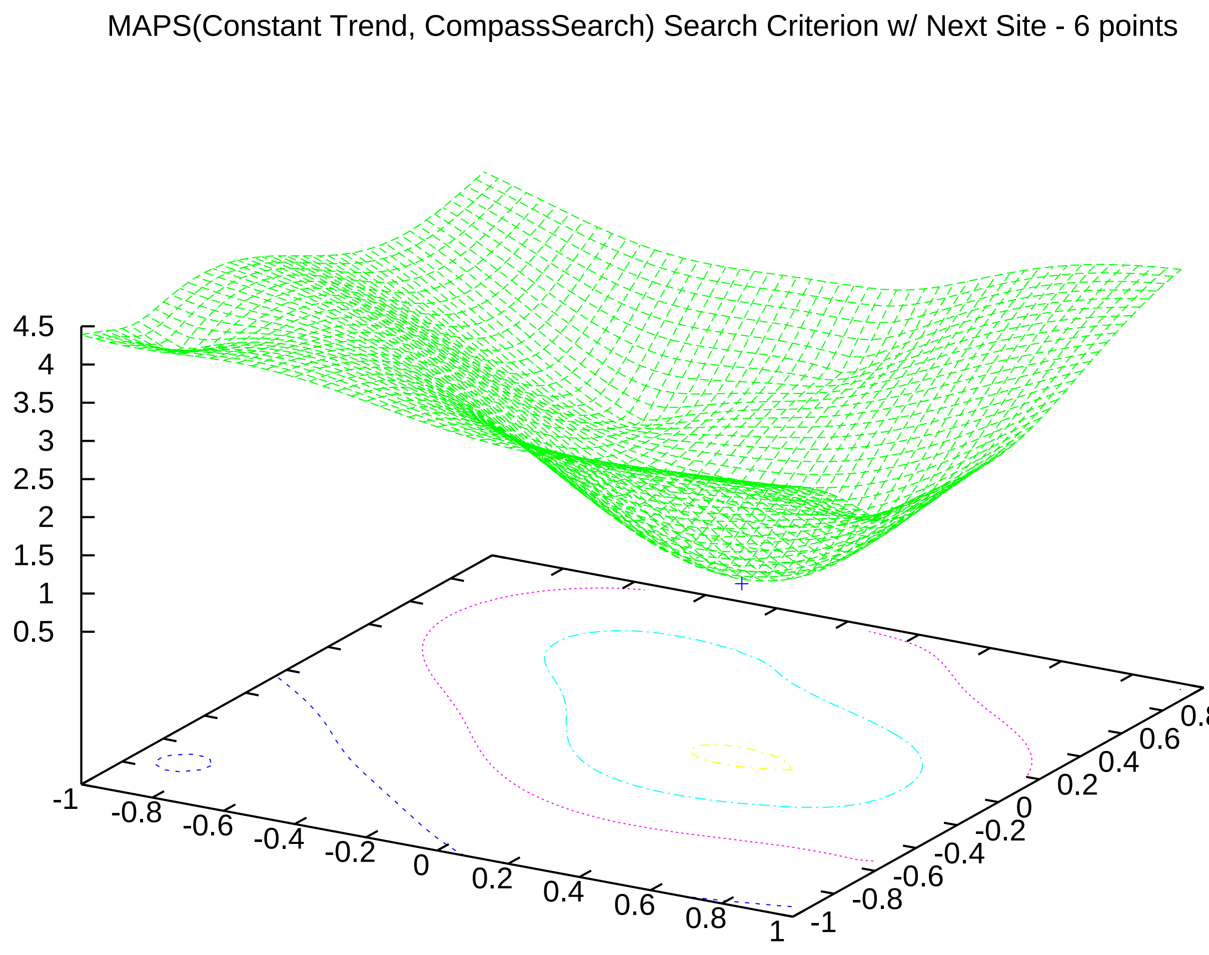
The first update of the approximation $\hat{f}_1$



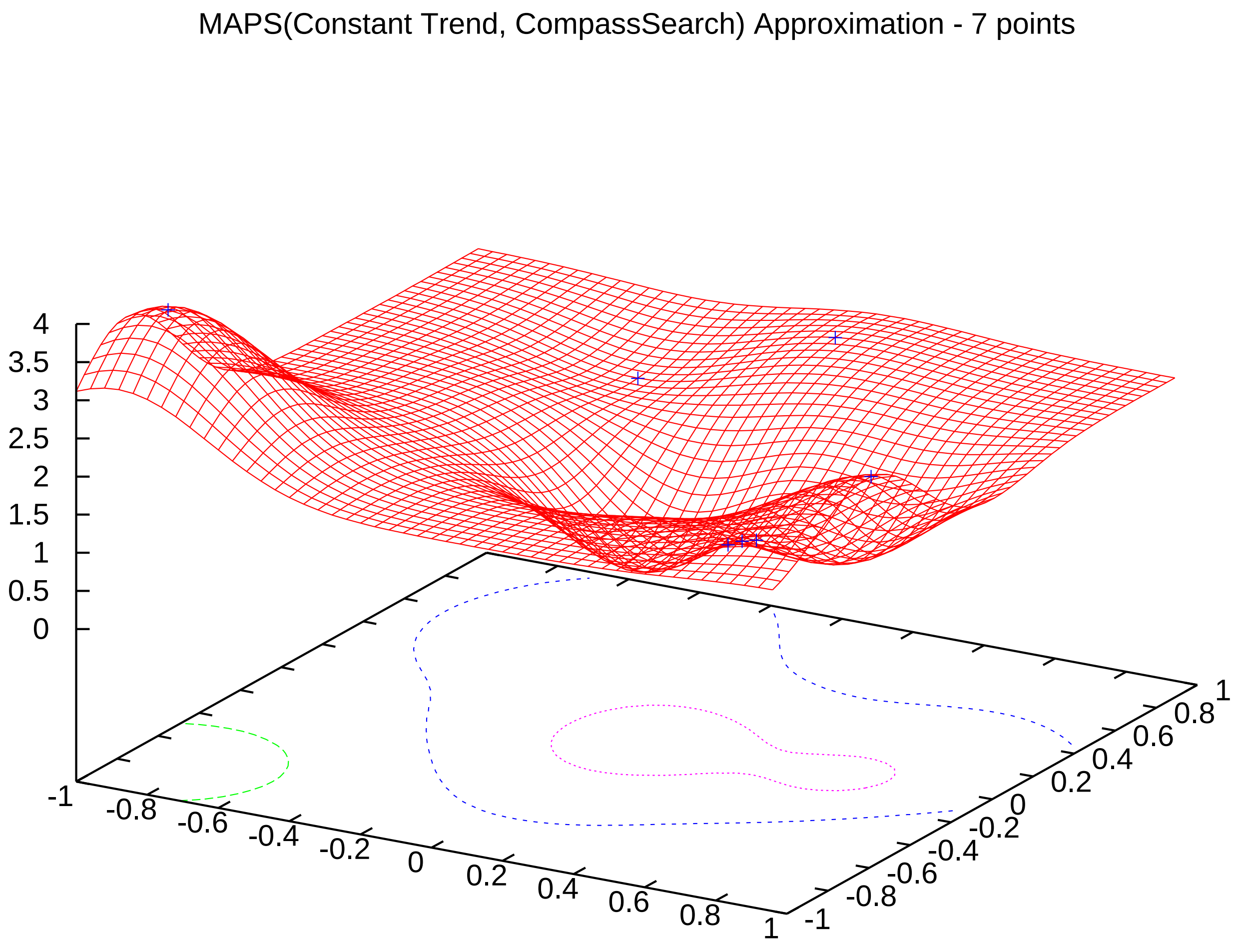
The objective f and the first update of the approximation $\hat{f}_1$



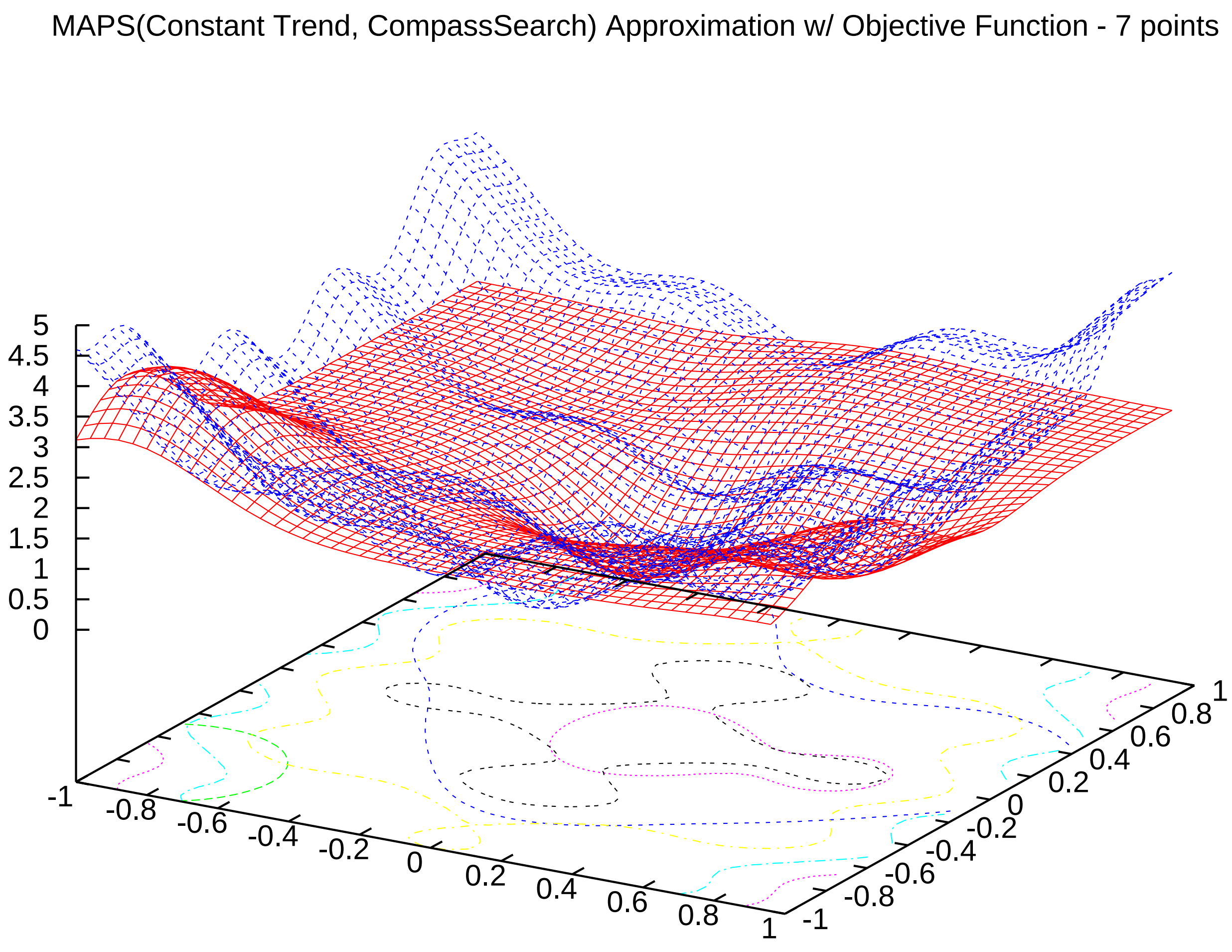
The first update of the search criterion S_1



The second update of the approximation $\hat{f}_2$



The objective f and the second update of the approximation $\hat{f}_2$



GSS methods for linearly constrained optimization

Return to to the the modified Broyden tridiagonal function:

$$\underset{x \in \mathbb{R}^2}{\text{minimize}} \;\; f(x^1, x^2)$$

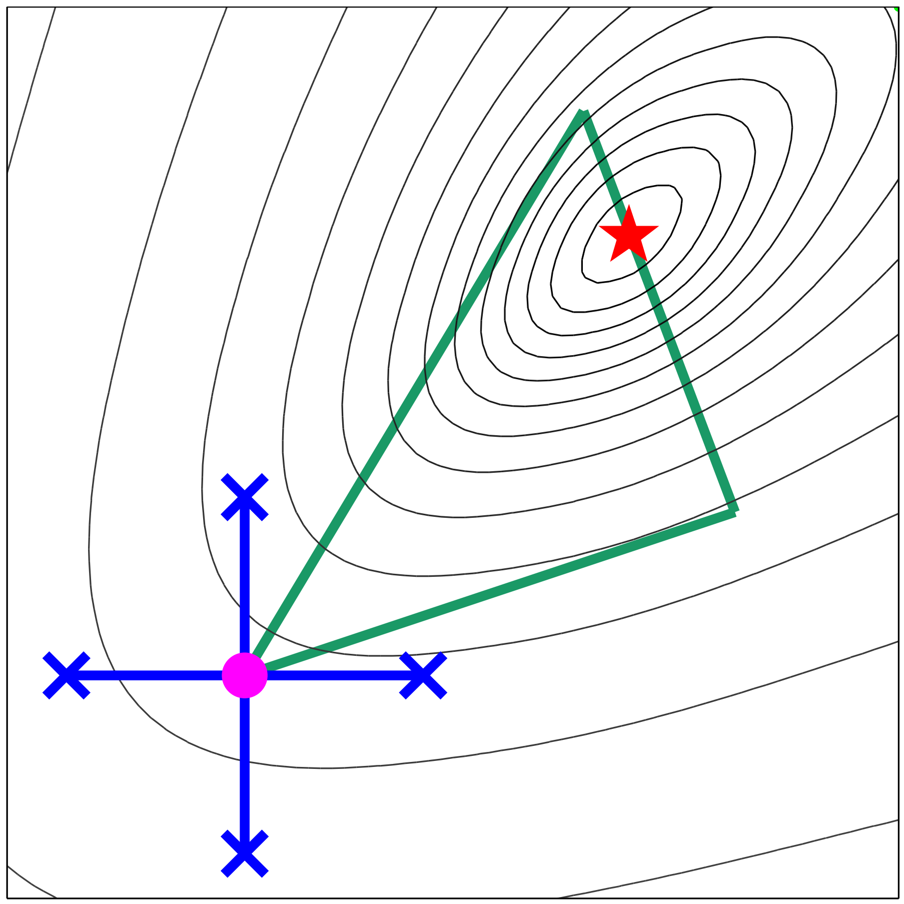
where

$$f(x) = \left|(3 - 2x^1)x^1 - 2x^2 + 1\right|^{\frac{7}{3}} + \left|(3 - 2x^2)x^2 - x^1 + 1\right|^{\frac{7}{3}},$$

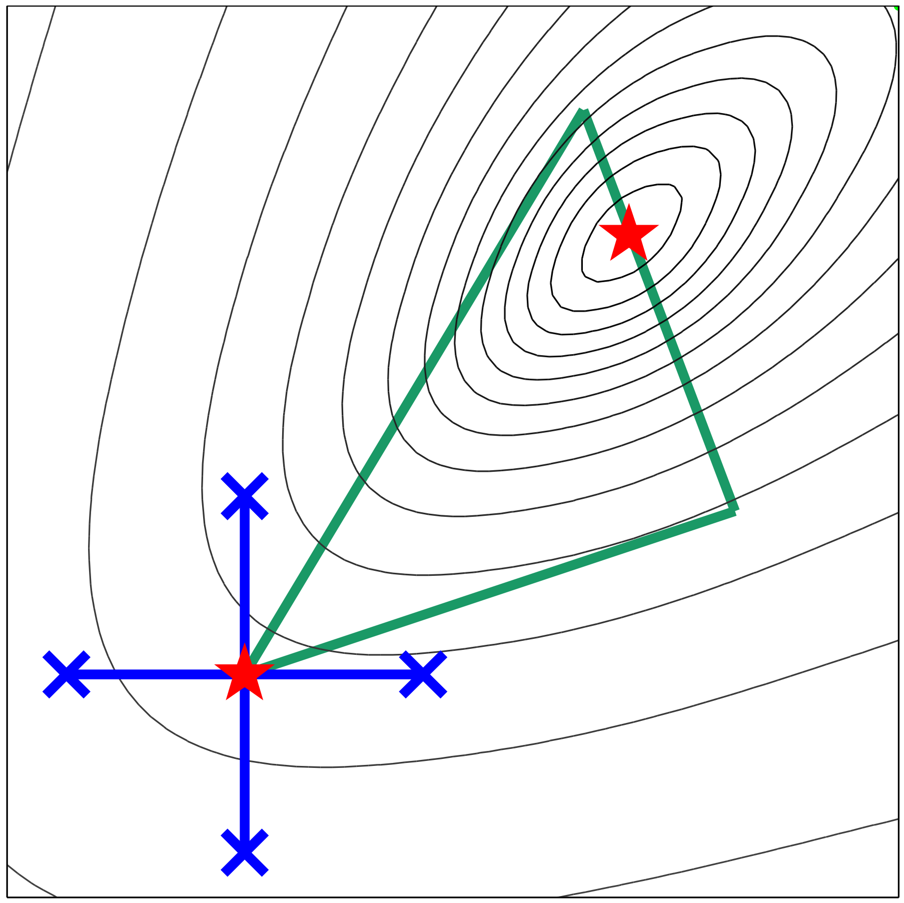
—now augmented with three linear constraints.

Again use a *feasible iterates* approach.

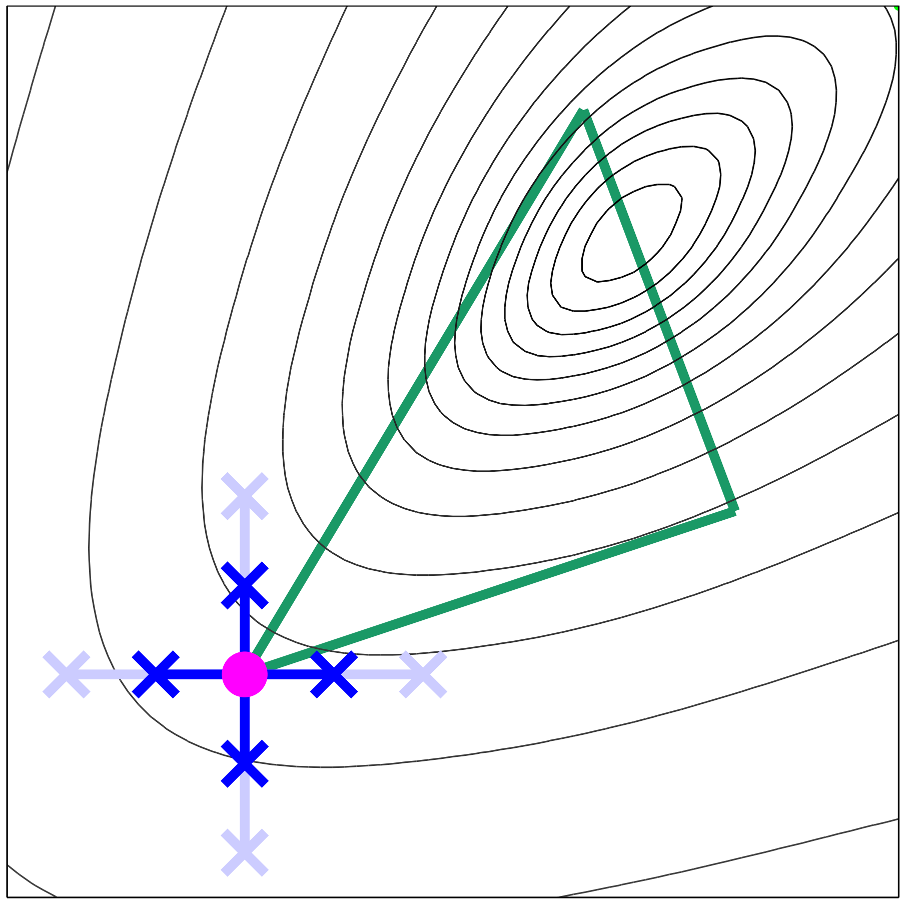
Start with the same initial configuration:



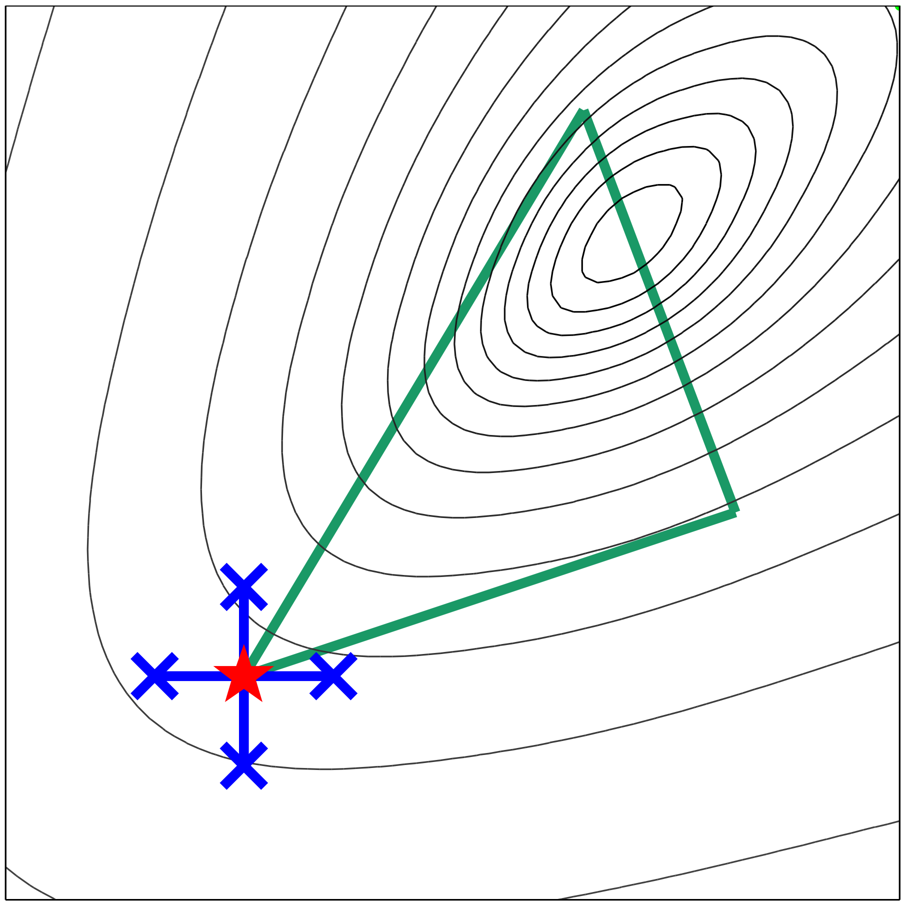
No feasible improvement identified



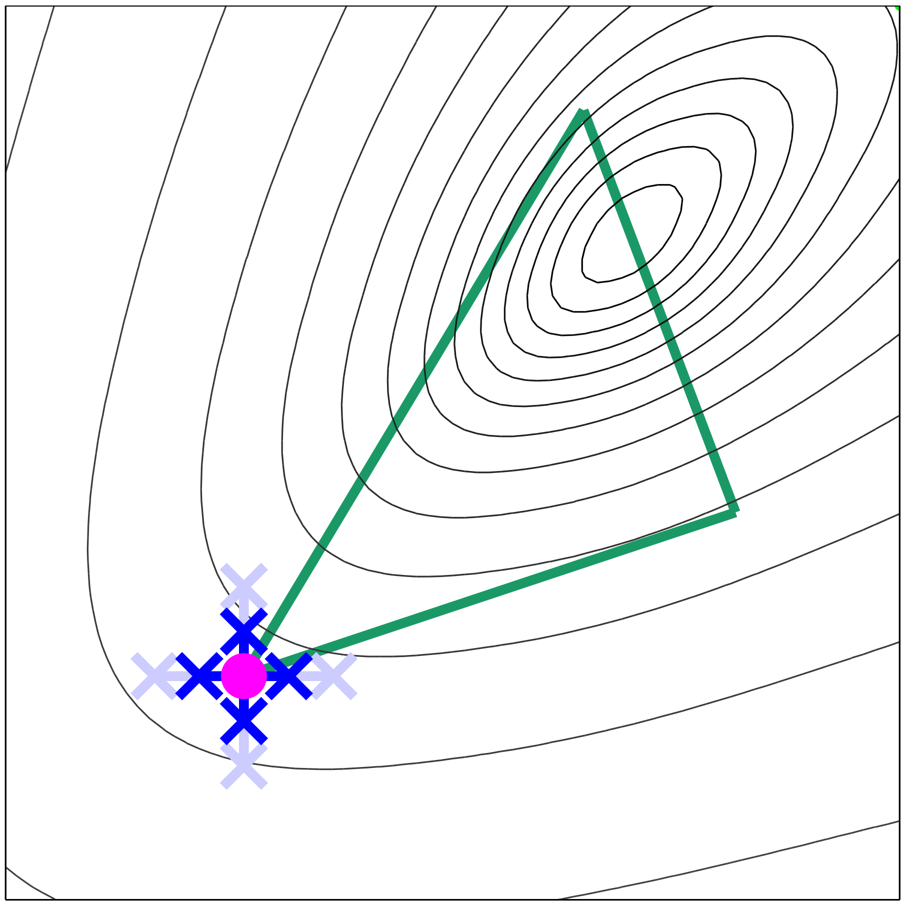
Contract; $k \in \mathcal{U}$



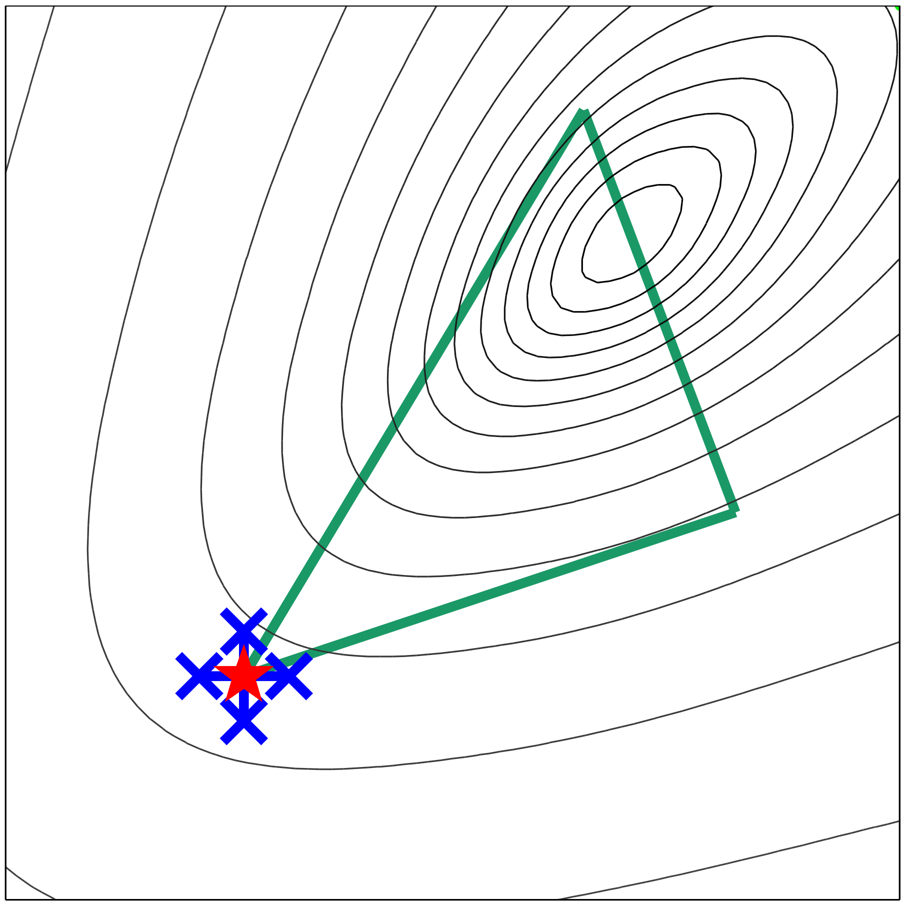
No feasible improvement identified



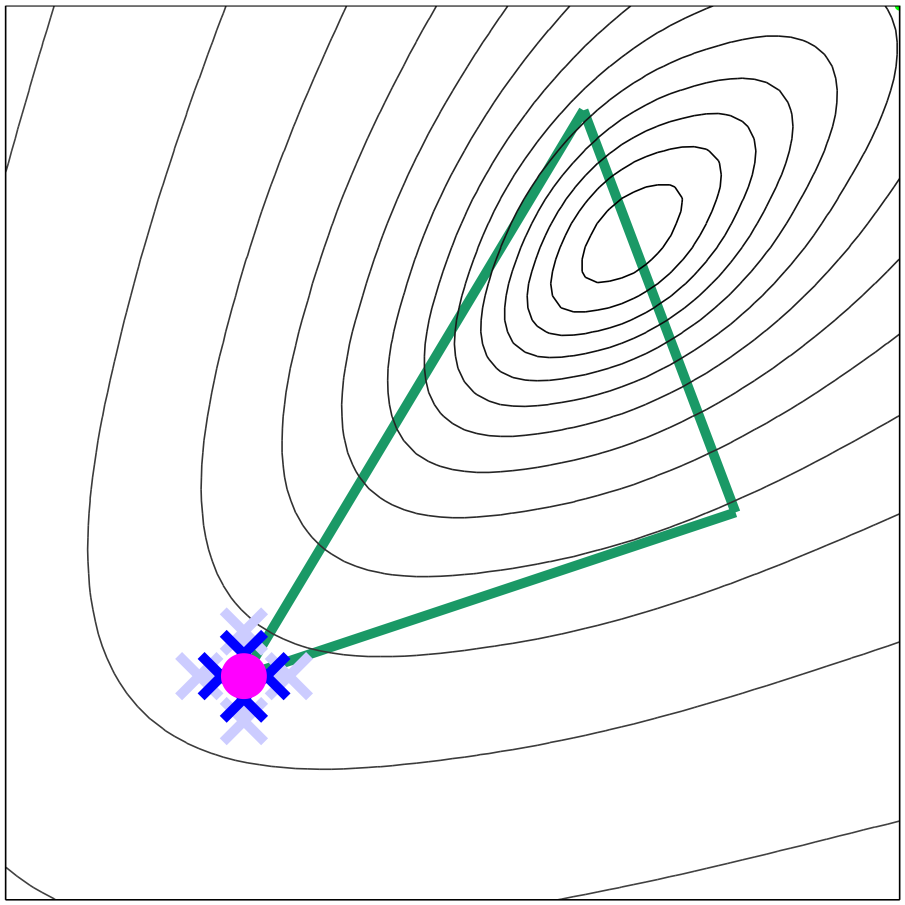
Contract; $k \in \mathcal{U}$



No feasible improvement identified



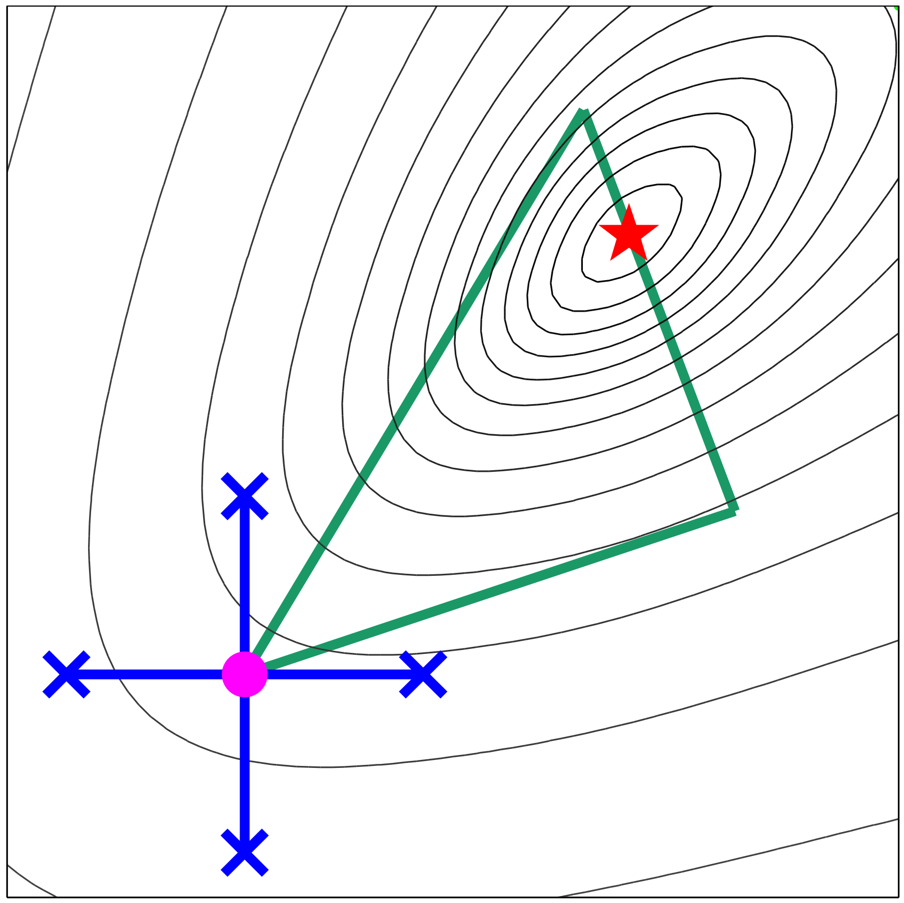
Contract; $k \in \mathcal{U}$....



Oops!!! The problem:

No *feasible* direction of descent.

Doomed from the start with this configuration:

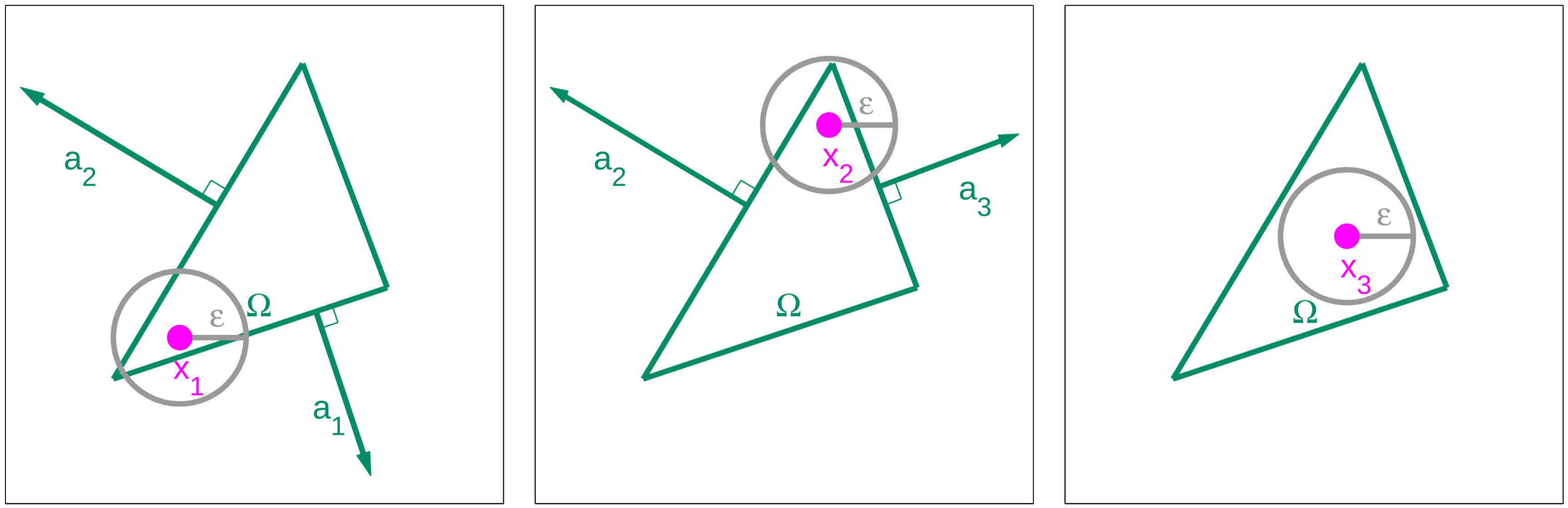


The fix:

Choose a set $\mathcal{G}_k$ that ensures feasible directions along the “nearby” constraints.

Identifying the nearby constraints

Find the outward-pointing normals within distance ε of the current iterate.



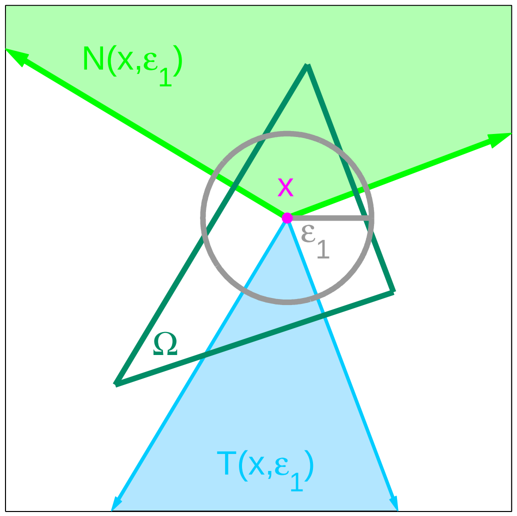
The conditions on ε depend on the convergence analysis in effect.

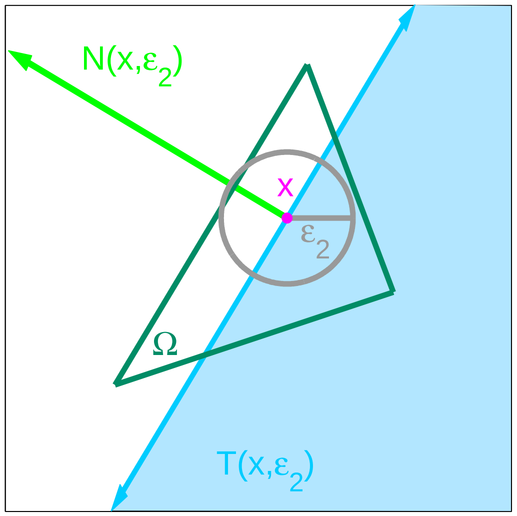
We use results from Kolda/Lewis/Torczon, 2006.

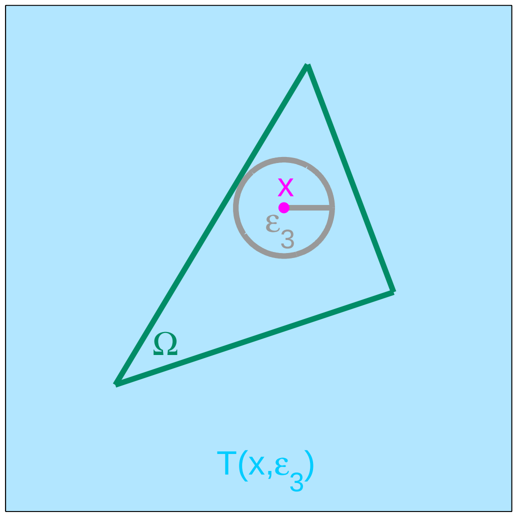
Obtaining a set of search directions: Part I

Translate the outward-pointing normals within distance ε of the current iterate x to obtain

- the ε-*normal cone* $N(x, \varepsilon)$ and
- its polar, the ε-*tangent cone* $T(x, \varepsilon)$.



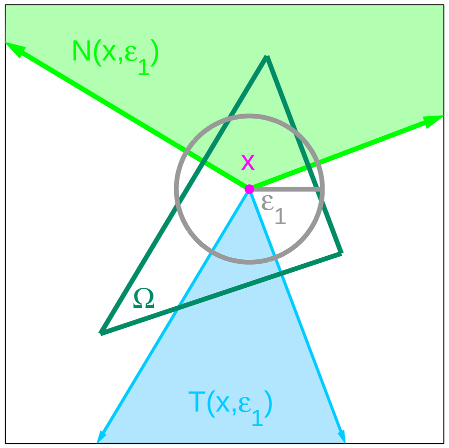


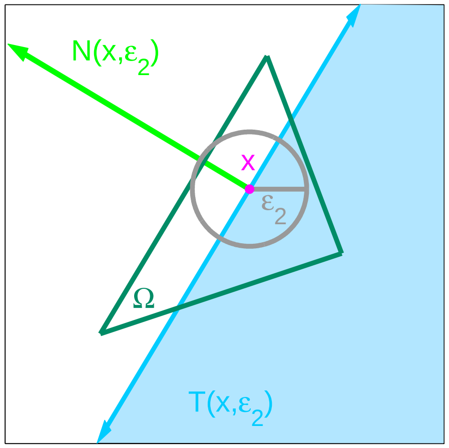


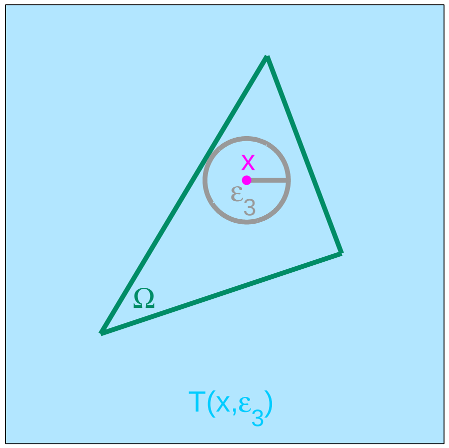
Critical observation:

- If $\varepsilon > 0$, then the set $x + T(x, \varepsilon)$ approximates the feasible region near x, where "near" is in terms of ε.

- If $T(x, \varepsilon) \neq \{\mathbf{0}\}$, then the search can proceed from x along all directions in $T(x, \varepsilon)$ for a distance of at least ε and still remain inside the feasible region.

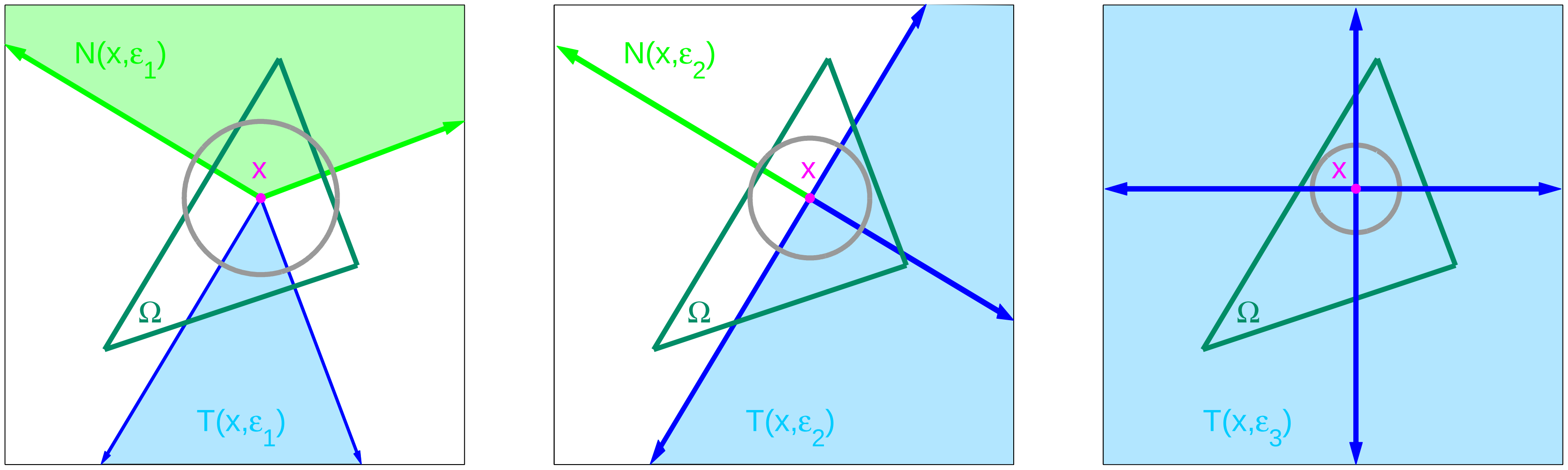






Obtaining a set of search directions: Part II

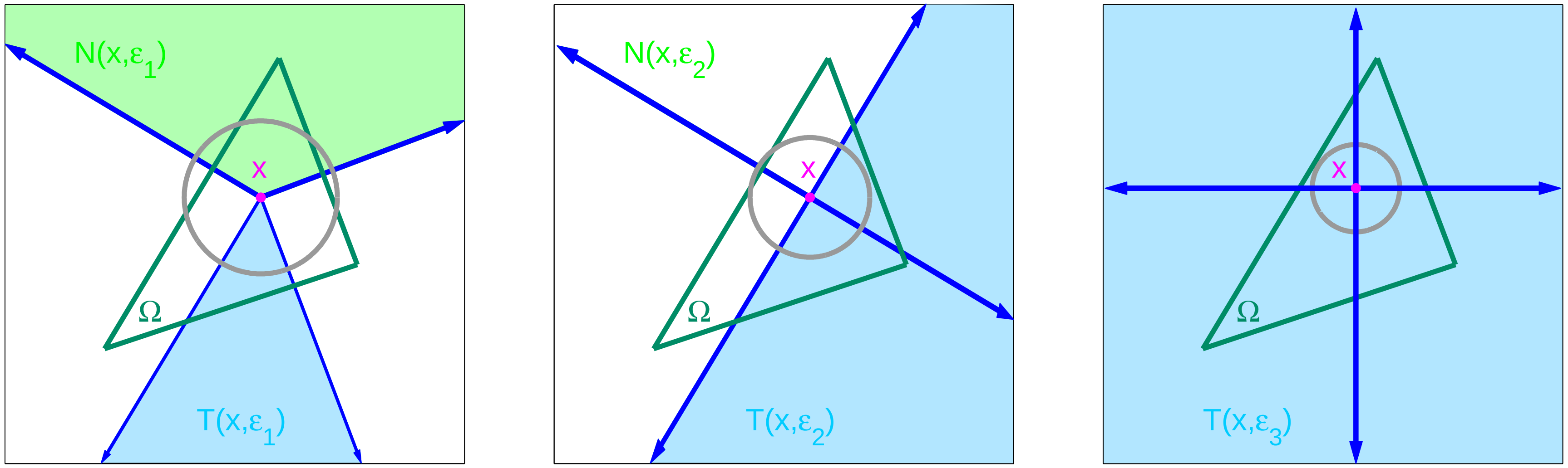
Require that the set $\mathcal{G}_k$ consist of generators for the ε-tangent cone $T(x, \varepsilon_k)$.



We use results from Lewis/Torczon, 1999 and Kolda/Lewis/Torczon, 2006.

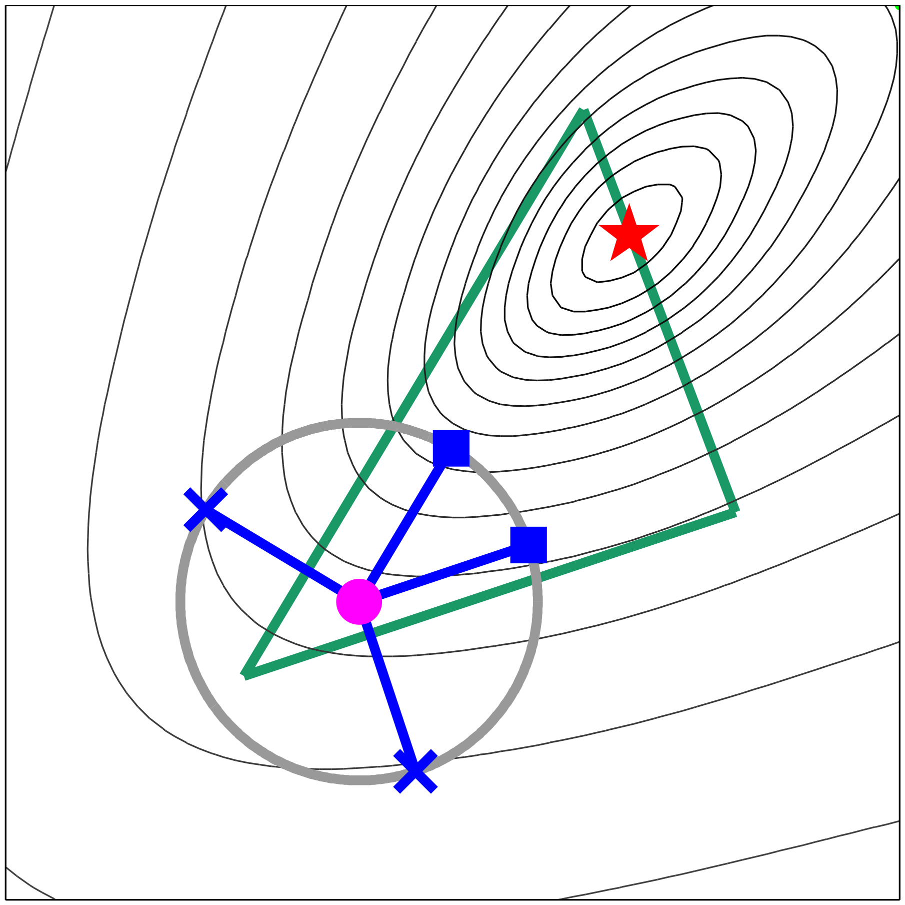
Obtaining a set of search directions: Part III

As a practical matter, include the set of generators for the ε-normal cone $N(x, \varepsilon_k)$ in the set of search directions $\mathcal{D}_k$.

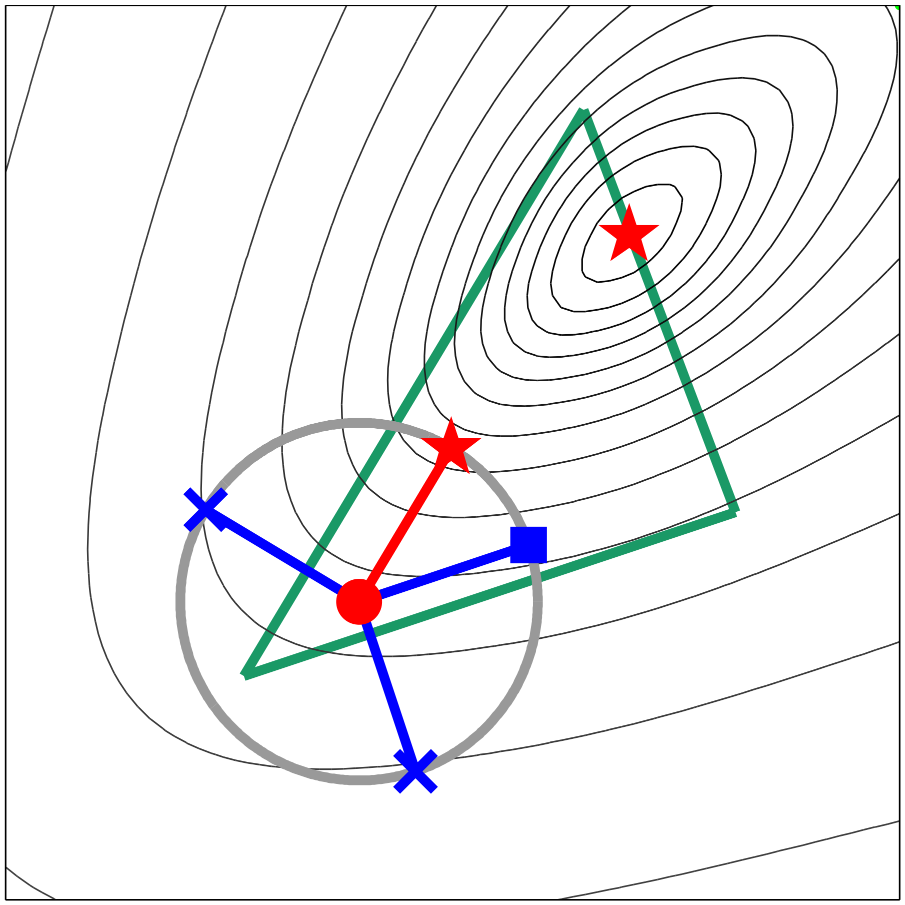


This necessarily means that $\mathcal{D}_k$ is a positive spanning set for $\mathbb{R}^n$.

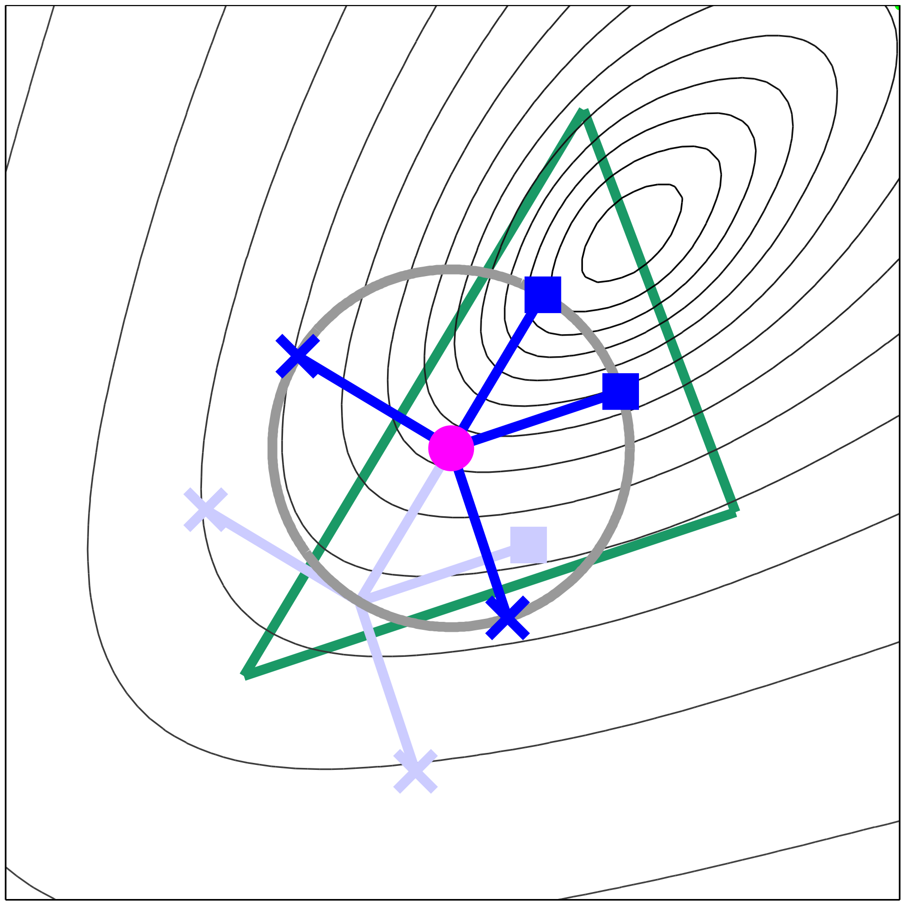
Returning to our example with the initial configuration:



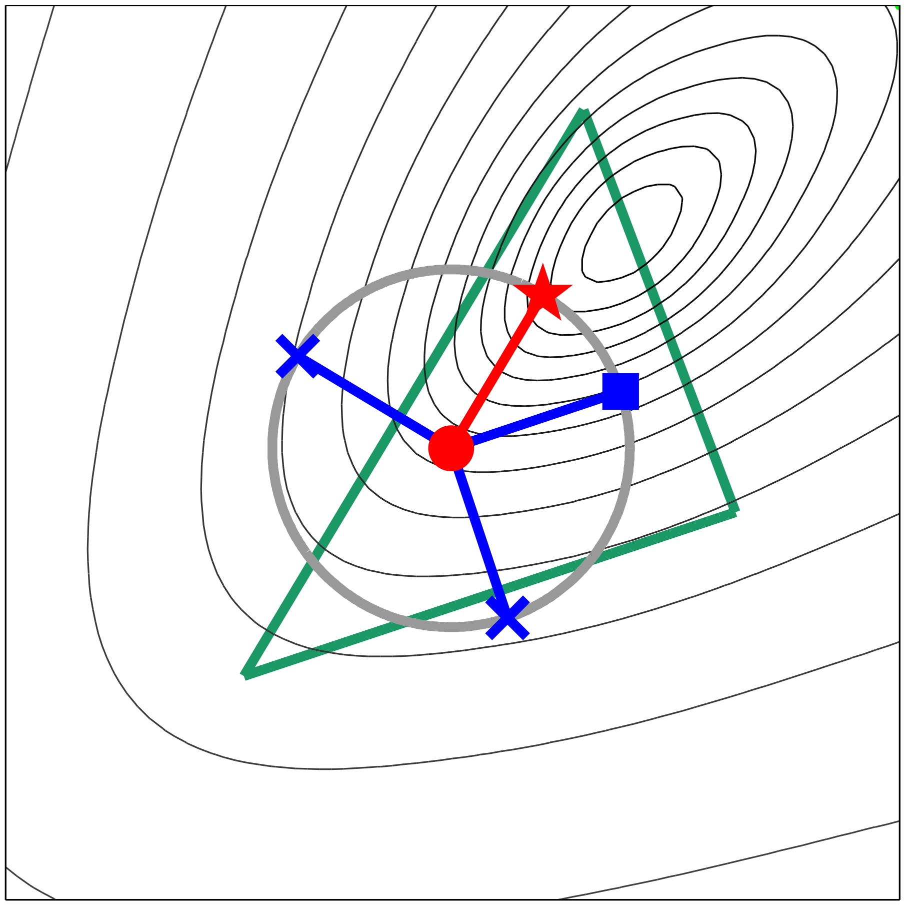
Identify feasible improvement:



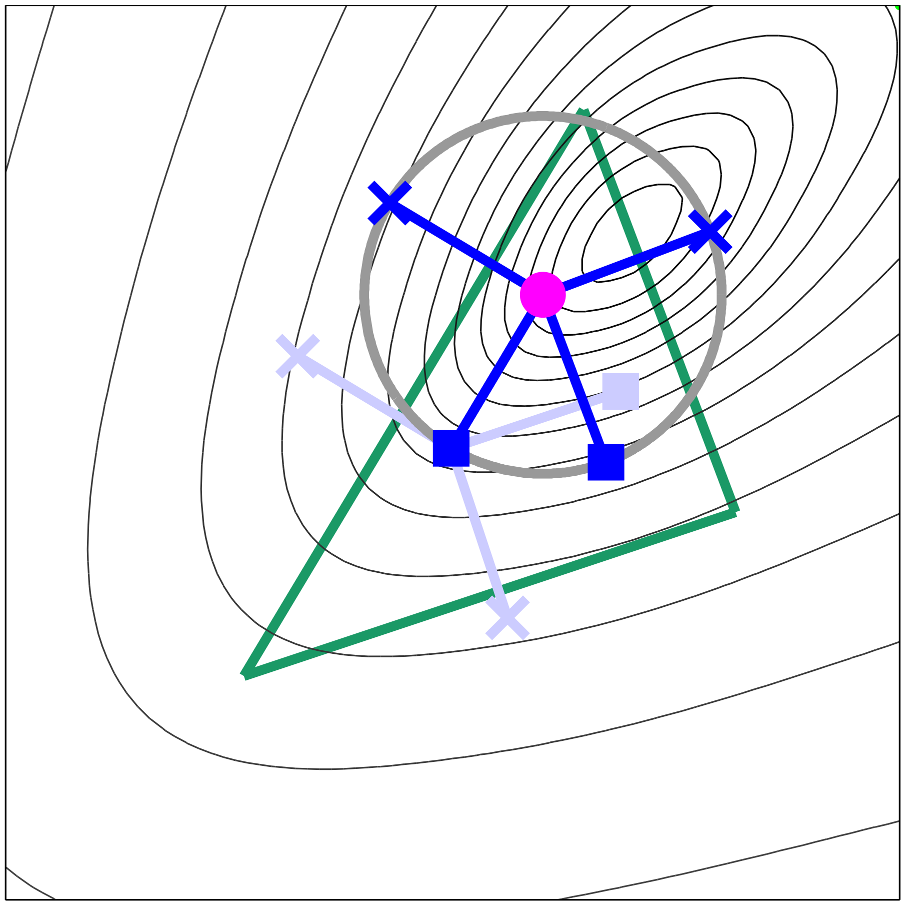
Move Northeast and keep the set of search directions



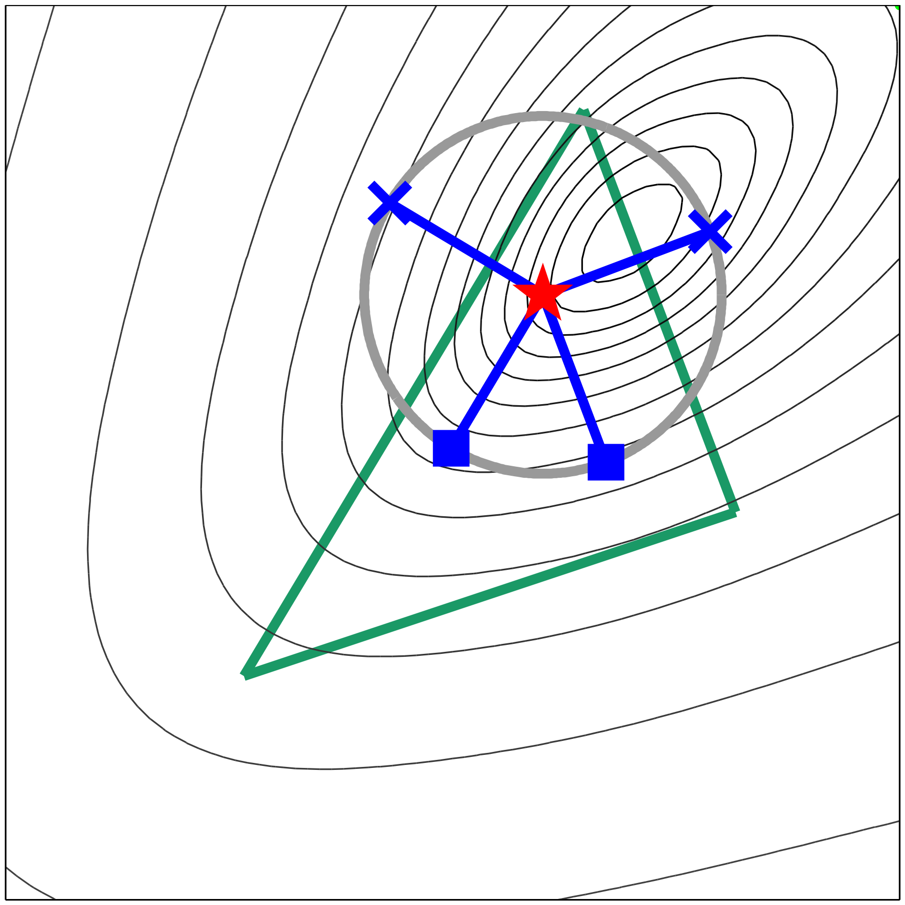
Identify feasible improvement:



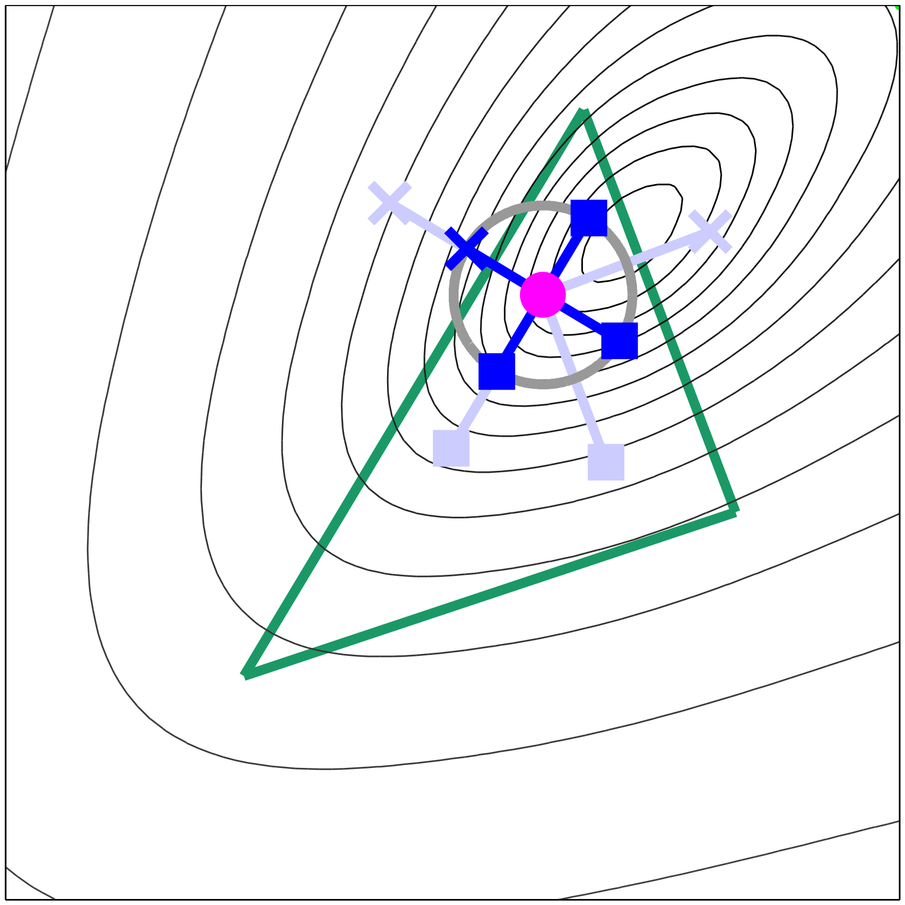
Move Northeast and *change* the set of search directions



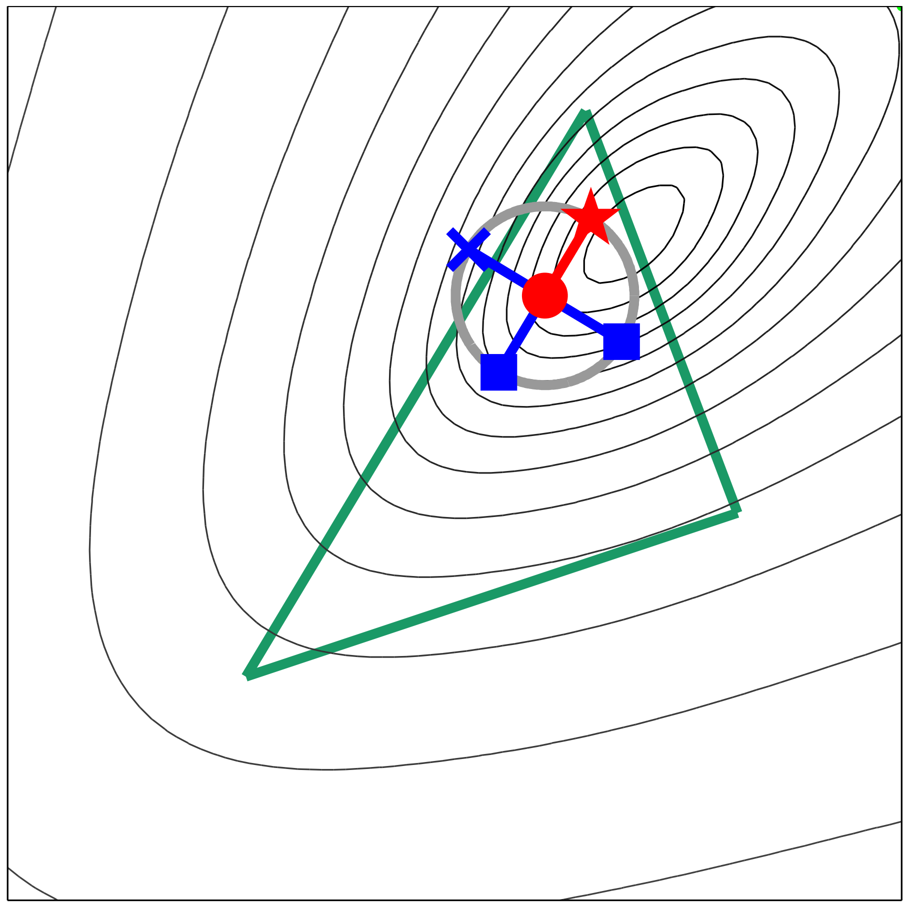
No feasible improvement:



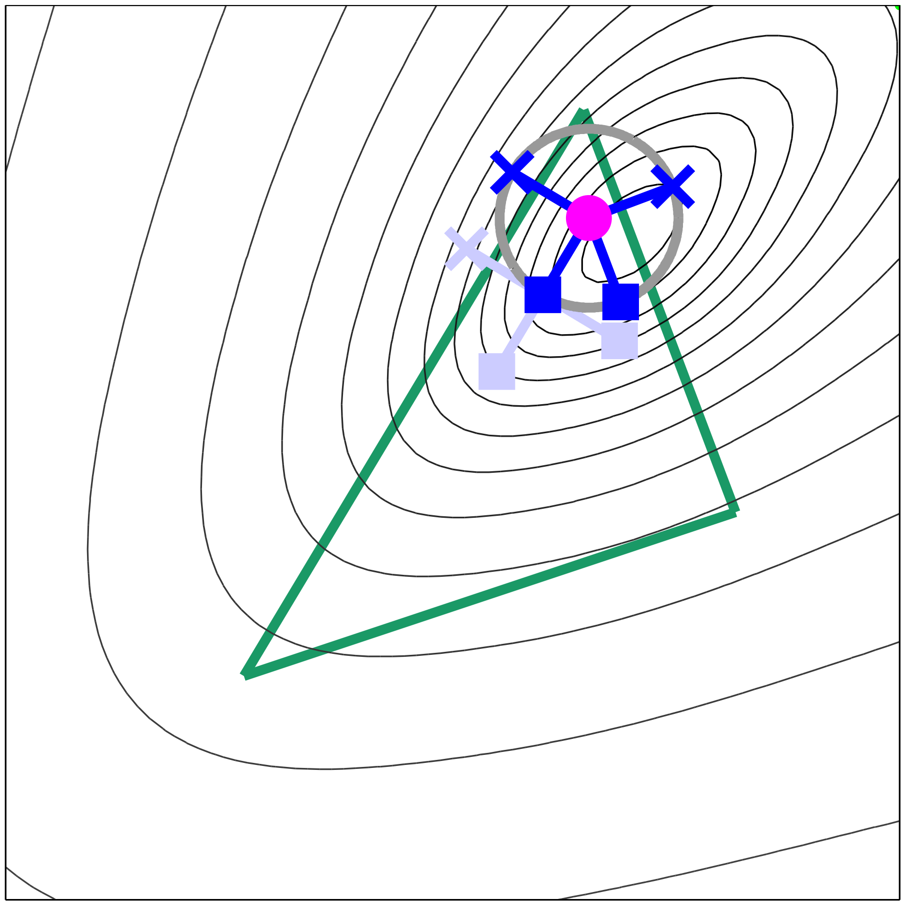
Contract and *change* the set of search directions



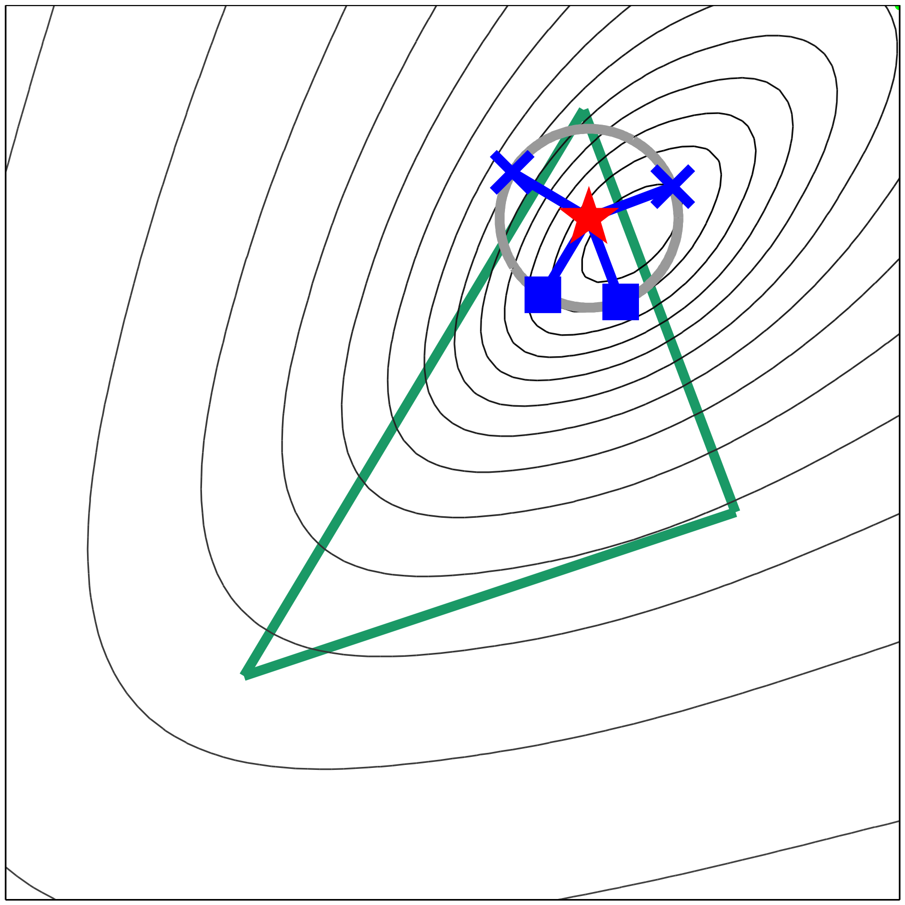
Identify feasible improvement:



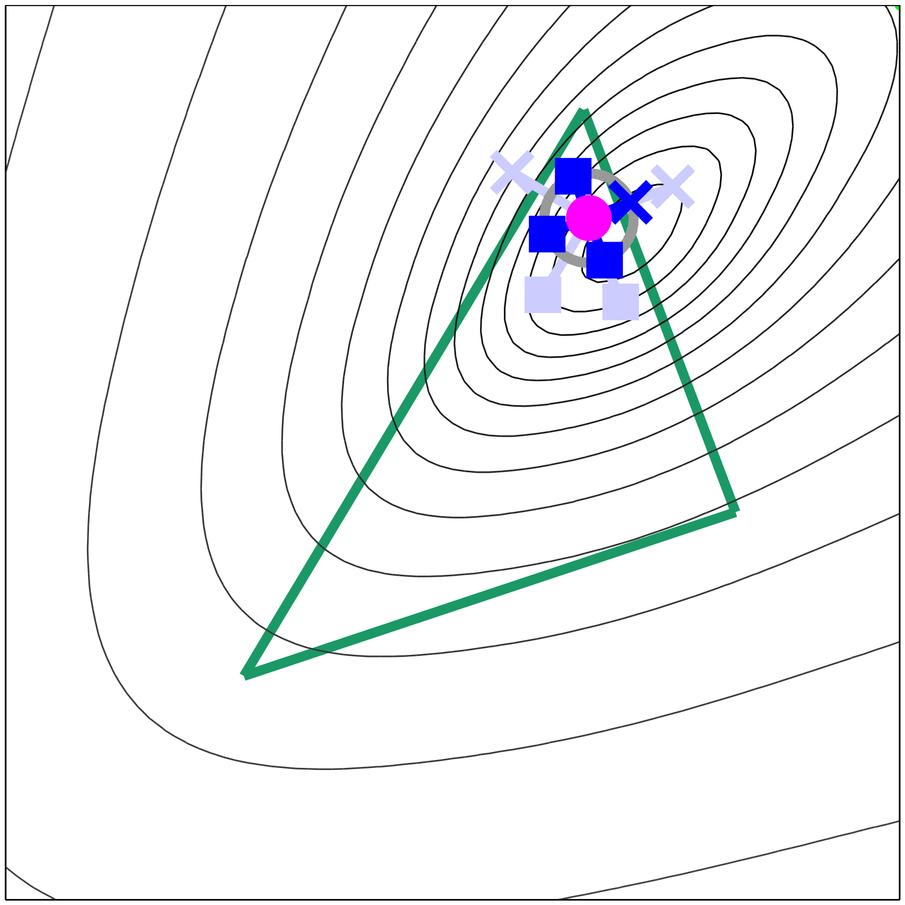
Move Northeast and *change* the set of search directions



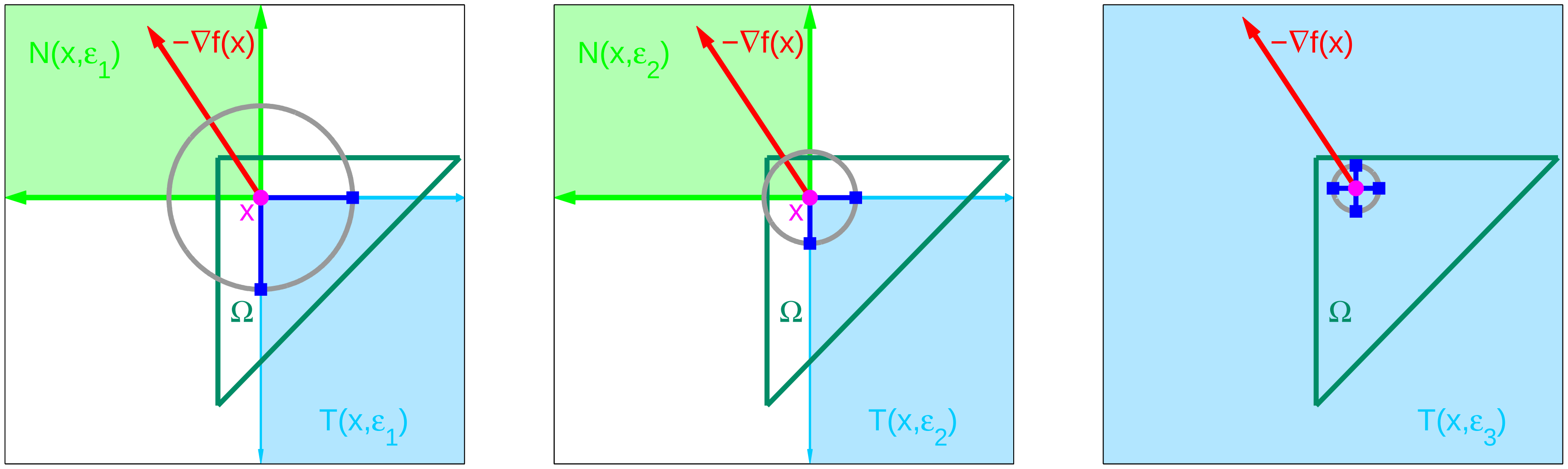
No feasible improvement:



Contract and *change* the set of search directions

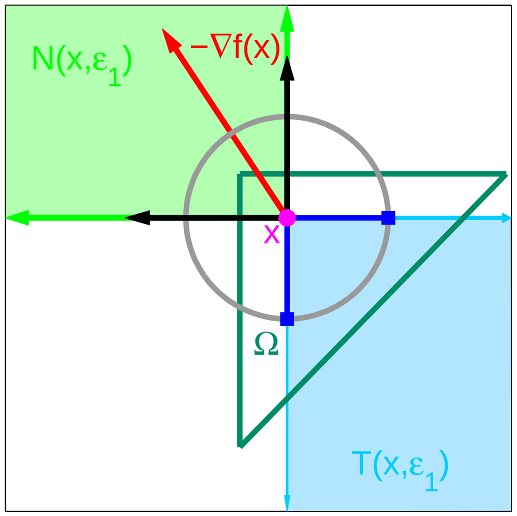


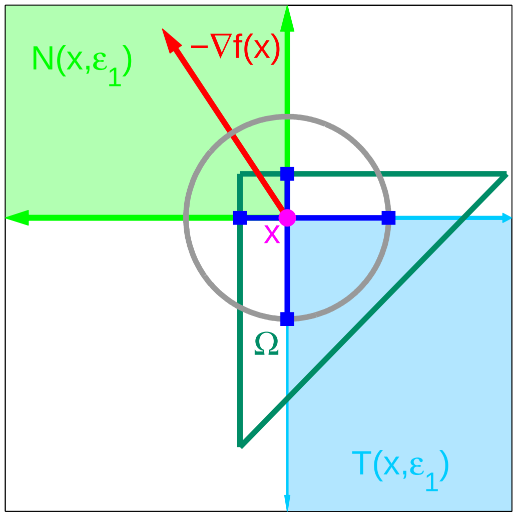
Why augment the set of directions and allow $c_k^i \, \Delta_k \, d_k^i$:



For this example, the minimalist approach requires at least 5 function evaluations (over three iterations) to identify a better point.

Why augment the set of directions and allow $c_k^i \, \Delta_k \, d_k^i$:





If, instead, include the generators of $N(x_k, \varepsilon_k)$ in $\mathcal{D}_k$—and allow exact steps to the boundary—may take as few as three function evaluations to identify a better point.

Minimal requirements for GSS methods for linear minimization:

- the set of search directions $\mathcal{D}_k$ include a generating set $\mathcal{G}_k$ for the ε-tangent cone $T(x_k, \varepsilon_k)$.

- the step-length control parameter Δ_k is reduced only when no feasible descent is identified for the step of length Δ_k along the directions $g_k^i \in \mathcal{G}_k$.

Certification for GSS methods for linearly constrained optimization is equivalent to that for the unconstrained case, with an appropriately chosen measure for linearly-constrained stationarity.

Flexibility in devising GSS methods for linearly constrained minimization:

- At each iteration the set of search directions $\mathcal{D}_k$ may include directions in addition to the set $\mathcal{G}_k$; e.g., $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$.

- For any $d_k^i \in \mathcal{D}_k$, it is possible to consider steps of the form $c_k^i \, \Delta_k \, d_k^i$.

- So long as $f(x_k + c_k^* \, \Delta_k \, d_k^*) < f(x_k) + \rho(\Delta_k)$ (i.e., sufficient improvement on $f(x_k)$ is found) and $(x_k + c_k^* \, \Delta_k \, d_k^*) \in \Omega$ (i.e., the step is feasible), either $d_k^* \in \mathcal{G}_k$ or $d_k^* \in \mathcal{H}_k$ is acceptable.

- The direction d_k^* need not be a *feasible descent direction*.

This admits heuristics for both acceleration schemes and global optimization.

The general nonlinear programming problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & c(x) = 0 \\ & Ax \geq b. \end{array}$$

Note:

- explicit linear constraints (including bounds) and
- general equalities with
- nonlinear inequalities converted to nonlinear equalities by introducing nonnegative slack variables.

Our approach to solving the general nonlinear programming problem:

Use an augmented Lagrangian approach adapted from by Conn, Gould, Sartenaer, and Toint (SIOPT, 1996) which involves

successive *linearly constrained minimization* of an augmented Lagrangian.

Goals:

- develop a deterministic generating set search algorithm and
- preserve the convergence properties of the original augmented Lagrangian algorithm.

Why leave the linear constraints explicit?

- If we have explicit linear constraints, then we have gradients for the explicit linear constraints: the rows a_i^T of the linear constraint matrix A.

- GSS methods can make effective computational use of this additional information and obtain good convergence behavior in both

  theory [Kolda, Lewis, and Torczon (SIOPT, 2006)] and
  practice [Lewis, Shepherd, and Torczon (SISC, to appear)].

General mantra for nonlinear programming: if you have problem structure to exploit, by all means do so!

One consequence of this handling linear constraints:

THEOREM 6.3 (Kolda/Lewis/Torczon, 2006) *Suppose that the gradient of f is Lipschitz continuous with constant M on the feasible region. Consider the linearly constrained GSS algorithms given in Kolda/Lewis/Torczon, 2006. If $k \in \mathcal{U}$ and ε_k satisfies $\varepsilon_k = \beta_{\max}\Delta_k$, then*

$$\| \, [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \, \| \leq \left(\frac{M\beta_{\max}}{\kappa_{\min}} \right) \Delta_k + \left(\frac{1}{\kappa_{\min}\beta_{\min}} \right) \frac{\rho(\Delta_k)}{\Delta_k},$$

where $\rho(\cdot)$ satisfies

$$\lim_{\Delta_k \downarrow 0} \frac{\rho(\Delta_k)}{\Delta_k} = 0.$$

This means

$$\| \, [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \, \| = O(\Delta_k).$$

Now let's look at the augmented Lagrangian approach

From Conn, Gould, Sartenaer, and Toint, given the original problem,

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & c(x) = 0 \\ & Ax \geq b, \end{array}$$

a classic solution technique is to minimize a suitable sequence of augmented Lagrangian functions. Including only the general equality constraints yields:

$$\Phi(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i c_i(x) + \frac{1}{2\mu} \sum_{i=1}^{m} c_i(x)^2,$$

where the components λ_i of the vector λ are the Lagrange multiplier estimates and μ is the penalty parameter.

Important:

The linear constraints $Ax \geq b$ are

- kept outside the augmented Lagrangian and
- handled at the level of the subproblem minimization,

thus allowing the use of specialized packages to solve linearly constrained problems.

Furthermore, the theory handles the linear inequality constraints in a purely geometric way. The same theory applies without modifications if linear equality constraints also are imposed and all the iterates are assumed to stay feasible with respect to these constraints.

The Conn, Gould, Sartenaer, and Toint inner iteration:

Find $x_k \in \mathcal{B} = \{x \,|\, Ax \geq b\} \neq \emptyset$ that approximately solves:

$$\min_{x \in \mathcal{B}} \Phi(x, \lambda_k, \mu_k) \equiv \Phi_k,$$

where the values of the Lagrangian multipliers λ_k and the penalty parameter μ_k are fixed for the subproblem.

By "approximately solved" it is meant that

$$\|P_{T(x_k,\omega_k)}(-\nabla_x \Phi_k)\| \leq \omega_k,$$

where $P_V(\cdot)$ is the projection onto the convex set V and ω_k is a suitable tolerance at iteration k.

Key “Ah, ha!”:

We do not have:

$$\|P_{T(x_k,\omega_k)}(-\nabla_x \Phi_k)\| \leq \omega_k.$$

But using our linearly constrained GSS algorithms means we do have:

$$\| \, [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \, \| = O(\Delta_k).$$

for *any* sufficiently smooth function f—including Φ_k.

In addition, recall that we set ε_k equal to $\beta_{\max}\Delta_k$.

The Kolda, Lewis, and Torczon inner iteration:

Find $x_k \in \mathcal{B} = \{x \,|\, Ax \geq b\} \neq \emptyset$ that approximately solves:

$$\min_{x \in \mathcal{B}} \Phi(x, \lambda_k, \mu_k) \equiv \Phi_k,$$

where the values of the Lagrangian multipliers λ_k and penalty parameter μ_k are fixed for the subproblem.

By "approximately solved" it is meant that we stop the solution of the kth subproblem at an unsuccessful (inner) iteration $s \in \mathcal{U}$ for which

$$\Delta_{k,s} \leq \delta_k,$$

where $\delta_k \to 0$ is updated at each iteration k.

Our key result:

With the stopping criterion we have substituted, the asymptotic behavior of

$$\|P_{T(x_k,\omega_k)}(-\nabla_x \Phi_k)\|$$

is like its behavior in the original Conn, Gould, Sartenaer, and Toint algorithm.

$\Longrightarrow$ the convergence analysis for the original algorithm can be applied and the original proofs still hold (e.g., any limit point of the sequence x_* is a Karush–Kuhn–Tucker point—a first-order stationary point—for the general nonlinear programming problem).

Current and future work:

- Investigate multiple algorithmic options within the augmented Lagrangian framework. The devil is in the details, including:
  - Lagrange multiplier updates,
  - active set identification, and
  - assorted other tweaks for acceleration.
- Explore other approaches to solving general nonlinear programming problems in the absence of derivatives.
- Devise effective variants for distributed computation.

Conclusions:

- The analysis gives insight and enables the design of effective algorithms.
- The devil is in the details.
- Papers are available from `http://www.cs.wm.edu/~va/research/`
- Implementations are available from The MathWorks in the *Genetic Algorithm and Direct Search Toolbox* `http://www.mathworks.com/products/gads/`

Some references:

- Kolda, Lewis, and Torczon, *Optimization by direct search: new perspectives on some classical and modern methods*, SIREV, 2003.
- Kolda, Lewis, and Torczon, *Stationarity results for generating set search for linearly constrained optimization*, SIOPT, 2006.
- Lewis, Shepherd, and Torczon, *Implementing generating set search methods for linearly constrained minimization*, SISC, to appear.
- Kolda, Lewis, and Torczon *A generating set search augmented Lagrangian algorithm for optimization with a combination of general and linear constraints*, in revision.
- Conn, Gould, Sartenaer, and Toint, *Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality constraints and linear constraints*, SIOPT, 1996.