APM 523 Global Optimization Project

Matt Kinsinger, Hongjun Choi, Adarsh Akkshai

September 27, 2017

[Part 2] We can define a class of functions to test our global optimization algorithms:

$$f(x) = \frac{\sum\limits_{i=1}^{k} f(z_i) \prod\limits_{j \neq i} \|x - z_j\|^{\alpha_j}}{\sum\limits_{i=1}^{k} \prod\limits_{j \neq i} \|x - z_j\|^{\alpha_j}}$$

$x, z_j \in [0,1]^n,\ \alpha_j > 0,\ j = 1, ..., k.$

We will show that the stationary points of this function are the z_j, that the values $f(z_j)$ can be defined to be whatever we choose, and that the maximum and minimum values of the function are $\max\limits_j\{f(z_j)\}$ and $\min\limits_j\{f(z_j)\}$. Thus we can test a global optimization algorithm using this function because we can define as many stationary points as we like while knowing before hand what the maximum and minimum values are.

a) To find the staionary points of f we need to find which points make $\nabla f = zero$. Since $f : \mathbb{R}^n \to \mathbb{R}$ the gradient will be an n-vector. We will consider an arbitrary coordinate of ∇f, hence a one-dimensional derivative and find when it is zero. Let x_m be a coordinate of x and consider the numerator of $\frac{\partial f(x)}{\partial x_m}$. Recalling the quotient rule we will need to calculate

$$\frac{bottom \cdot \frac{\partial(top)}{\partial x_m} - top \cdot \frac{\partial(bottom)}{\partial x_m}}{bottom^2}$$

This calculation is done on the next page, we ignore the denominator since we are only concerned with finding where the partial derivative is zero.

$$bottom \cdot \frac{\partial(top)}{\partial x_m} - top \cdot \frac{\partial(bottom)}{\partial x_m}$$

$$\left[\sum_{i=1}^{k}\prod_{j\neq i}\|x-z_j\|^{\alpha_j}\right]\left[\sum_{i=1}^{k} f(z_i)\sum_{l\neq i}\left[\frac{\alpha_l}{2}\left(\|x-z_l\|^{\alpha_l}\right)^{-1}\cdot 2\left(x_m - z_{l,m}\right)\left(\prod_{p\neq i,l}\|x-z_p\|^{\alpha_p}\right)\right]\right]$$

$$-\left[\sum_{i=1}^{k} f(z_i)\prod_{j\neq i}\|x-z_j\|^{\alpha_j}\right]\left[\sum_{i=1}^{k}\sum_{l\neq i}\left[\frac{\alpha_l}{2}\left(\|x-z_l\|^{\alpha_l}\right)^{-1}\cdot 2\left(x_m - z_{l,m}\right)\left(\prod_{p\neq i,l}\|x-z_p\|^{\alpha_p}\right)\right]\right]$$

WLOG suppose that $x = z_1$, then the above equation reduces to

$$\left[\prod_{j\neq 1}\|z_1-z_j\|^{\alpha_j}\right]\left[f(z_1)\sum_{l\neq 1}\left[\frac{\alpha_l}{2}\left(\|z_1-z_l\|^{\alpha_l}\right)^{-1}\cdot 2\left(z_{1,m} - z_{l,m}\right)\left(\prod_{p\neq 1,l}\|z_1-z_p\|^{\alpha_p}\right)\right]\right]$$

$$-\left[f(z_1)\prod_{j\neq 1}\|z_1-z_j\|^{\alpha_j}\right]\left[\sum_{l\neq 1}\left[\frac{\alpha_l}{2}\left(\|z_1-z_l\|^{\alpha_l}\right)^{-1}\cdot 2\left(z_{1,m} - z_{l,m}\right)\left(\prod_{p\neq 1,l}\|z_1-z_p\|^{\alpha_p}\right)\right]\right]$$

$$= 0.$$

Since this holds for an arbitrary coordinate of $\nabla f(z_1)$ where z_1 was chosen WLOG, we can conclude that $\nabla f(z_j) = 0$ each j. Thus each z_j, j = 1,2,...k, is a stationary point of f.

Note: We are not certain that the vectors z_j are the only stationary points of f, however in part b) we conjecture a possible explanation of why it may be possible to ensure that the vectors z_j are in fact the only stationary points (although our reasoning is admittedly far from rigorous).

b) Consider f when we evaluate it at one of the z_j, say WLOG $f(z_1)$. Then we have

$$
\begin{aligned}
f(z_1) &= \frac{\sum\limits_{i=1}^{k} f(z_i) \prod\limits_{j \neq i} \|z_1 - z_j\|^{\alpha_j}}{\sum\limits_{i=1}^{k} \prod\limits_{j \neq i} \|z_1 - z_j\|^{\alpha_j}} \\
&= \frac{f(z_1) \prod\limits_{j \neq 1} \|z_1 - z_j\|^{\alpha_j}}{\prod\limits_{j \neq 1} \|z_1 - z_j\|^{\alpha_j}} \\
&= f(z_1).
\end{aligned}
$$

Thus we are free to choose $f(z_j)$ to be whatever we desire. In particular it may be possible to choose each $f(z_j)$ to be such that $\nabla f(x) \neq 0$ for all $x \neq z_j$, all j. This implies that the set $\{z_1, ..., z_k\}$ contains all of the stationary points of f. Furthermore, the maximum and minimum of f are $\max\limits_j f(z_j)$ and $\min\limits_j f(z_j)$, respectively and they are only one global maximal and minimal. Since we choose the values of $\{f(z_1), ..., f(z_k)\}$ we are able to know before hand what the maximum and minimum values are for this function which has k stationary points. Hence we have a function with multiple local extrema with knowledge of the true global min/max. That is why this works as a test function for global optimization algorithms.

c) Observing the terms in the derivation of the partial derivative $\frac{\partial f(x)}{\partial x_m}$ we see that we must have $\alpha_j \geq 2$ in order for f to be differentiable. This is to ensure that $\frac{\alpha}{2} - 1 \geq 0$, so that

$$
\left(\|x - z_l\|^{\alpha_l}\right)^{-1} = \left[\sum_{i=1}^{n} (x_i - z_{l,i})^2\right]^{\frac{\alpha_l}{2} - 1}
$$

does not grow without bound as $x \to z_l$.

d) Test cases using **simman.m**:

Case 1: $n = 2$

$k = 4$

$\{z_1,\ z_2,\ z_3,\ z_4\} = \{(.1,\ .7),\ (.33,\ .14),\ (.16,\ .3),\ (1,\ .5)\}$

$\{f(z_1),\ f(z_2),\ f(z_3),\ f(z_4)\} = \{7, -3, -2, 5\}$

$\{\alpha_1,\ \alpha_2,\ \alpha_3,\ \alpha_4\} = \{3.5,\ 4.5,\ 4.1,\ 2.7\}$

$x_0 = \{.5,\ .5\}$

Minimization of f: $x_{opt} = \{.33,\ .14\} = z_2$, $f_{opt} = -3.0 = f(z_2)$.

Maximization of f: $x_{opt} = \{1,\ .7\} = z_1$, $f_{opt} = 7.0 = f(z_1)$.

Case $1'$: Same as Case 1, maximization, but with $\alpha_1 = .5 < 2$, so that f is not differentiable at z_1. The results are:

x_{opt} = {-1,-1} (a boundary point of f)
$f_{opt} = 5.05$

f is not differentiable at z_1, hence z_1 can no longer be a minimum value of f since it is not a stationary point of f.

Case 2: $n = 3$
$k = 4$
$\{z_1,\ z_2,\ z_3,\ z_4\}$ = {(.1, .7, .6), (.75, .14, .9), (.6, .1, .7), (1, 0, .5)}
$\{f(z_1),\ f(z_2),\ f(z_3),\ f(z_4)\}$ = {4, -1, 3, 7}
$\{\alpha_1,\ \alpha_2,\ \alpha_3,\ \alpha_4\}$ = {2.5, 3, 4.1, 2.7}
x_0 = {.5, .5, .5}

Minimization of f: $x_{opt} = \{.75,\ .14,\ .9\} = z_2$, $f_{opt} = -1.0 = f(z_2)$.
Maximization of f: $x_{opt} = \{1,\ 0,\ .5\} = z_4$, $f_{opt} = 7.0 = f(z_4)$.

Case $2'$: Same as Case 2, minimization, but with $\alpha_2 = 1.5 < 2$, so that f is not differentiable at z_2. The results are:

x_{opt} = {-1,-1,-1} (a boundary point of f)
$f_{opt} = 1.797$

Just as in Case $1'$, f is not differentiable at z_2, hence z_2 can no longer be a minimum value of f since it is not a stationary point of f.

(See the directory **part2d** for the matlab scripts and outputs related to question 2d)

[Part 3] We are comparing the capabilities of the AMPL IPOPT global solver versus the matlab FMINCON, *interior-point algorithm* solver. Our comparison will be in terms of computational time (measured in seconds). The function we are trying to minimize is

$$f(x) = \sum_{i=1}^{n-1} \left[\left(x_i^2\right)^{\left(x_{i+1}^2+1\right)} + \left(x_{i+1}^2\right)^{\left(x_i^2+1\right)} \right], \qquad x \in \mathbb{R}^n.$$

Below is a tablulation of our results:

Solver	time n=5000	time n=10000	relevant files	
AMPL IPOPT	0.12	0.28	brown.mod brown5000.out	brown.cmd brown10000.out
FMINCON (no gradient)	691.1	989.9	brownf.m brownf5000.out	brownf10000.out
FMINCON (gradient)	223.2	685.2	brownfg.m brownfg5000.out	brownfg10000.out
FMINCON (grad and Hessian)	22.5	82.7	brownfgH.m brownfgH5000.out	hessianfcn.m brownfgH10000.out

Clearly, the AMPL IPOPT is superiour to matlab's FMINCON for solving global optimization problems.

Remark: These were some of the problems we encountered when trying to supply the derivative (browfg.m):

i. The supplied derivative (calculated in brownfgh.m) and the forward finite difference derivative (the approximation calculated by fmincon.m) were not within the default tolerance of 10e-6. This caused an error in the algorithm. This comparison is only done once in the algorithm for a randomly perturbed point x. It is meant as a safegaurd to check that the user has calculated the supplied derivative correctly. After inspection we realized that the calculated derivative was only slightly outside of this tolerance. We changed the algorithm's derivative approximation method to "center finite difference" (more computationally expensive, but more accurate) and the problem was resolved.

ii. The returned objective value was in the order of 10e87 (overflow had occured). One of our first attempts at fixing this was to decrease the "step size tolerance" so that the algorithm would have to do more iterations, but this was not relevant to the problem and was unsuccessful. We came to realize (after a short interaction with Dr. Mittleman) that our objective function f growing out of control was due to our x value moving beyond some point which caused f to become overly large and cause overflow. We needed to place artificial bounds on x to keep this from happening. We forced x to remaim within the interval [-10,10] and this solved the problem.

Problems when trying to supply the hessian (brownfgH.m):

i. We needed to create a new function dedicated to supplying the hessian. We solved this problem by copying the hessian evaluation section from brownfgh.m to create hessianfcn.m, and then placing 'HessianFcn',@hessianfcn in the options for FMINCON.

   (see directory **part3** for relevant scripts and outputs)

[Part 4] **We placed this question at the end of the document.**

[Addition]

– On p.191 of the book we are told that the extended Rosenbrock function

$$f(x)=\sum_{i=1}^{n/2}\left[\alpha\left(x_{2i}-x_{2i-1}^2\right)^2+(1-x_{2i-1})^2\right],\qquad n\text{ even, }\alpha>0.$$

has a minimizer at $x^*=(1,1,...,1)^T$. We need to prove that this is the only minimizer of f. We do this by computing both $\frac{\partial f}{\partial x_m}$ and $\frac{\partial f}{\partial x_{m+1}}$ where $1\le m\le n-1$ is odd. If we can show that setting both of these equal to zero forces $x_m=x_{m+1}=1$ then we have proven that the only possible minimizer of f is $x^*=(1,1,...,1)^T$.

$$\frac{\partial f(x)}{x_m}=-4\alpha x_m\left(x_{m+1}-x_m^2\right)-2(1-x_m) \tag{1}$$

$$\frac{\partial f(x)}{x_{m+1}}=2\alpha\left(x_{m+1}-x_m^2\right) \tag{2}$$

Setting (2) equal to zero we force

$$\left(x_{m+1}-x_m^2\right)=0 \qquad (3)$$

Considering (3), and setting (1) equal to zero forces

$$1-x_m=0$$

$$x_m=1$$

Which, by (3), implies that

$$x_{m+1}=1.$$

Thus, $x^*=(1,1,...,1)^T$ is the only minimizer of f. □

1) Show that the Rosenbrock function is not convex in $\mathbb{R}^4$ by computing the Hessian at (-1/2,1/2,1/2,1/2) and finding its eigenvalues in matlab. The Hessian is

$$\left\{\begin{array}{cccc}-400x_2+1200x_1^2+2 & -400x_1 & 0 & 0\\ -400x_1 & 202-400x_3+1200x_2^2 & -400x_2 & 0\\ 0 & -400x_2 & 202-400x_4+1200x_3^2 & -400x_3\\ 0 & 0 & -400x_3 & 200\end{array}\right\}$$

Evaluated at (-1/2,1/2,1/2,1/2) gives us

$$\left\{\begin{array}{cccc} 102 & 200 & 0 & 0 \\ 200 & 302 & -200 & 0 \\ 0 & -200 & 302 & -200 \\ 0 & 0 & -200 & 200 \end{array}\right\}$$

When we use matlab's **eig.m** function to find the eigenvalues of this matrix we have

$$\begin{aligned} \lambda_1 &= & -81.56 \\ \lambda_2 &= & 62.57 \\ \lambda_3 &= & 331.26 \\ \lambda_4 &= & 593.72 \end{aligned}$$

showing that f has a non-positive-definite Hessian matrix at (-1/2,1/2,1/2,1/2). Hence f is not convex.

2) We will use an AMPL script with IPOPT to show that f has a second local minimizer near (-1,1,...,1) for $n > 4$.

n	x_0	x^*
5	(-1,1,..,1)	(-.962,.936,.881,.778,.605)
10	(-1,1,..,1)	(-.993,.997,.998,....,.988)
20	(-1,1,..,1)	(-.993,.997,.998,....,.9999)

This shows that f has a local minimizer near (-1,1,...,1) and that it is increasingly closer to (-1,1,...,1) as n increases.

(see directory **addition** for the relevant AMPL scripts and outputs)

3) We did not finish this question.

[Part 4]

In geometry and coding theory, a **spherical code** with parameter (n, N, t) is a set of N points on the unit hypersphere in n dimensions for which the dot product of unit vectors from the origin to any two points is less than or equal to t.

The kissing number problem (KNP) is defined as the number of non-overlapping unit spheres that an be arranged such that they each touch another given unit sphere. In general, the KNP seeks the maximum possible kissing numbers N for a n dimensional spheres in $(n+1)$ dimensional Euclidean space. The KNP may be stated as the problem of finding the maximal N for a given n such that a spherical code with parameters $(n, N, 1/2)$ exists.

THe KNP highlights the problem statement, and set of parameters we can play around to design optimization problem. Our goal is the Tammes problem,

here we pack circles onto the surface of the sphere for a given set of points. One fixes the number of circles and then makes them as large as possible without overlapping. In other words, one tries to place points on the surface of the sphere that have maximal distance from each other. The Tammes problem may be stated as the problem of finding spherical code with minimal t for given n and N.
In summary, the Tammes problems can be referred to as how we can place N points on a spheres in the n-dimensional space in order to maximize the minimal distance between any two points on spheres. Or formally speaking;

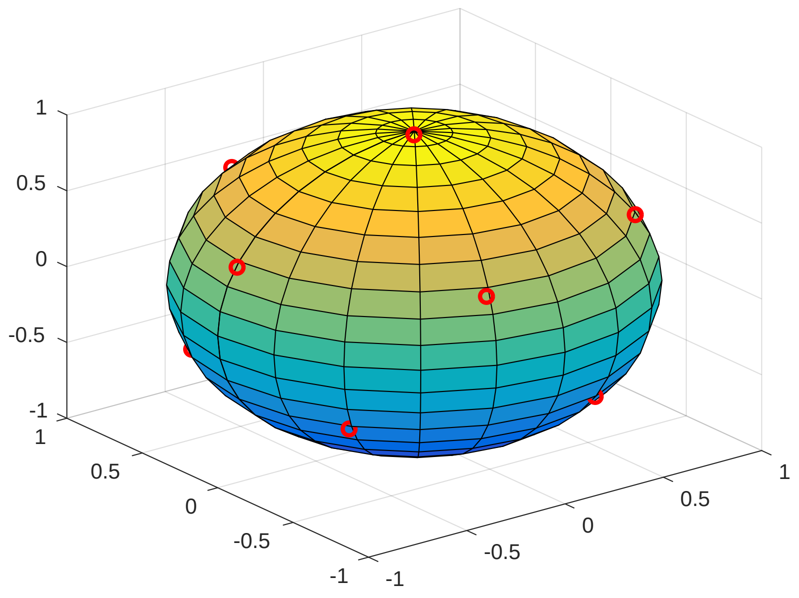
$$E_T = \max_{\|x_i\|=1, 1\le i\le N} \min_{1\le j<k\le N} \|x_j - x_k\|$$

where $x_i \in \mathbb{R}^n$ and the vector norm $\|\cdot\|$ is the Euclidean norm.
We are going to code that in AMPL (our script in **tammes.mod**) and obtain the center points of each circle as large as possible without overlapping by proposing proper constraints. Without loss generality, the spheres is normalized to a radius of 1. In our situation we are in $\mathbb{R}^3$ and a integer $N = 12$ points.
We solved the Tammes problem by using NEOS-solvers **KNITRO** by implementing the methods above within the framework. Our computational results were obtained as below tables.

Name	Results
Final objective value	2.51692658297786e-09
Final feasibility error (abs / rel)	2.22e-16 / 2.22e-16
Final optimality error (abs / rel)	5.15e-14 / 5.15e-14
The number of iterations	8929
The number of CG iterations	3826
The number of function evaluations	27613
The number of gradient evaluations	8939
The number of Hessian evaluations	8929
Total program time (secs)	17.20079

The co-ordinates of the given points on the sphere are as indicated in the below table.

We used Matlab to plot these points on the sphere to ascertain if these points are visually correct.



1
0.5
0
-0.5
-1
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

X	Y	Z
-0.708023	0.54763	-0.445876
-0.0282906	-0.0399966	0.998799
0.28467	-0.833764	-0.473076
0.0282906	0.0399966	-0.998799
-0.90671	-0.0123963	0.421573
-0.713289	-0.502971	-0.488096
0.708023	-0.54763	0.445876
-0.28467	0.833764	0.473076
0.90671	0.0123963	-0.421573
-0.293192	-0.866144	0.404762
0.713289	0.502971	0.488096
0.293192	0.866144	-0.404762

Lastly, we computed the values between different points and verified if the angle is the same as one measured in Wikipedia. To this end we calculate the given angle using the vector angle formula in 3-D.

$$\theta = \arccos\left(\frac{\langle A, B\rangle}{\|A\|.\|B\|}\right)$$

where A and B are the two vectors from the origin to the centers of the circles and $\|\cdot\|$ is the ℓ_2-norm.

To prove out point, here we show one such computed angle with respect to point 6 in the table.

Points	Angle
1	63.4350
2	116.5650
3	63.4349
4	63.4350
5	63.4350
6	0.0913
7	116.5650
8	116.5651
9	116.5650
10	63.4350
11	179.9087
12	116.5650

Clearly the points 1,3,4,5,10 are points adjacent to point 6 as per our visual and algebraic evaluation. The Euclidean distance between these two points is 1.703.