Chapter 1

Smoothing Techniques

1.1 Reasons for Smoothing

Sometimes the function is very nasty and has many local optima, so it is very difficult for Trust-Region or Simulated Annealing to find the global one. The smoothing technique can help simulated annealing to find the global optimum more efficiently. By efficiently, we mean take a little longer time and get a much better answer.

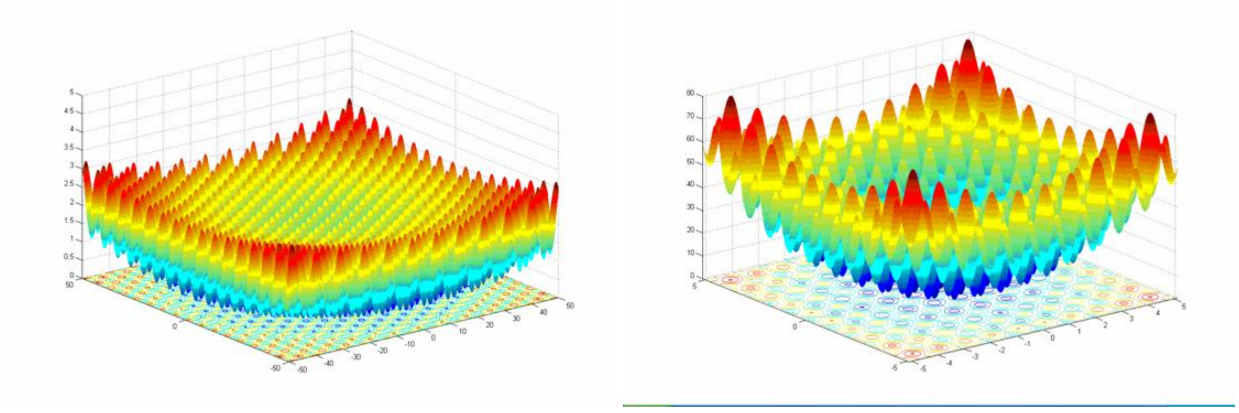


Figure 1.1: Griewank and Rastrigin function

1.2 Two Ways for Smoothing

- $\bar{f}(x) = f(x) + \frac{1}{6}\Delta^2 trace(H)$
- $\bar{f}(x) = f(x) + \lambda||x - x_*||_2^2$

Where H is the Hessian matrix and x_* is the global optimum we guess. Δ and λ is defined by user.

We call the first one *Trace-Smooth* and the second one λ-*Smooth*

Now we state the derivation of the formula: $\bar{f}(x) = f(x) + \frac{1}{6}\Delta^2 trace(H)$.

Let f be an objective function and Δ be a positive number. The average value of f over a regular Δ-box $Box(x)$ centred at x with sides $[x_i - \Delta, x_i + \Delta]$ is:

$$\bar{f}(x) = \frac{1}{(2 * \Delta)^n} \int_{Box(x)} f(x) dx_1 ... dx_n \tag{1.1}$$

The formula above is too expensive to compute when n is large or function f is difficult to compute. However, by approximating f using quadratic Taylor series expansion

$$f(x + s) \cong f(x) + g^T s + \frac{1}{2} s^T H s \equiv q(x) \tag{1.2}$$

where $g = \nabla f(x)$, $H = \nabla^2 f(x)$,1.1 can be approximated as

$$\bar{f}(x) \cong \bar{q}(x) = f(x) + \frac{1}{(2 * \Delta)^n} \int_{\forall i, |s_i| \leq \Delta} (g^T s + \frac{1}{2} s^T H s) ds_1 ... ds_n \tag{1.3}$$

Since

$$g^T s + \frac{1}{2} s^T H s = \sum_i g_i s_i + \sum_i \sum_j s_i s_j h_{ij} \tag{1.4}$$

Interchanging the order of summation and integration of the above formula yields:

$$\bar{f}(x) = f(x) + \frac{1}{6}\Delta^2 \cdot trace(H) \tag{1.5}$$

1.3 The Pictures of $Trace - Smooth$ Technique

Here we give 4 examples for using the smoothing technique $\bar{f}(x) = f(x) + \frac{1}{6}\Delta^2 \cdot trace(H)$.

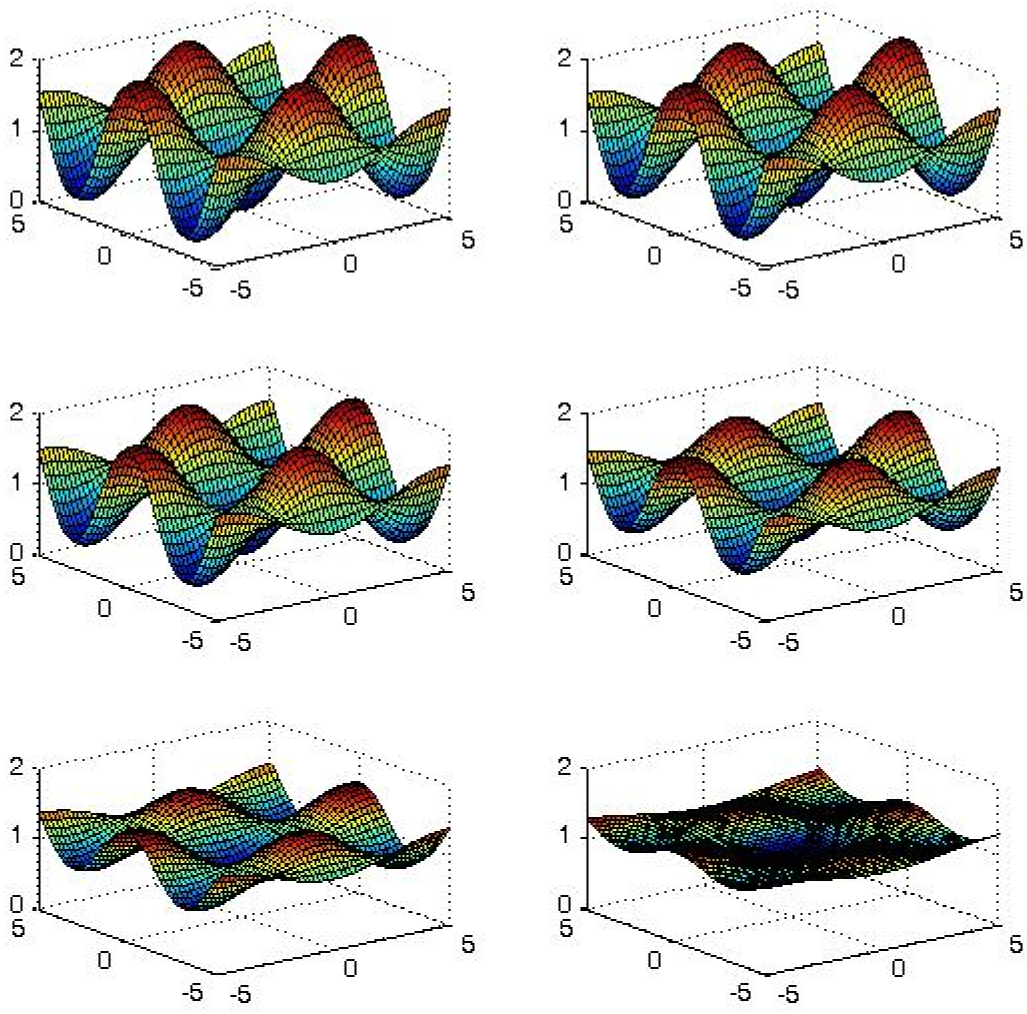


Figure 1.2: Griewank Smoothing

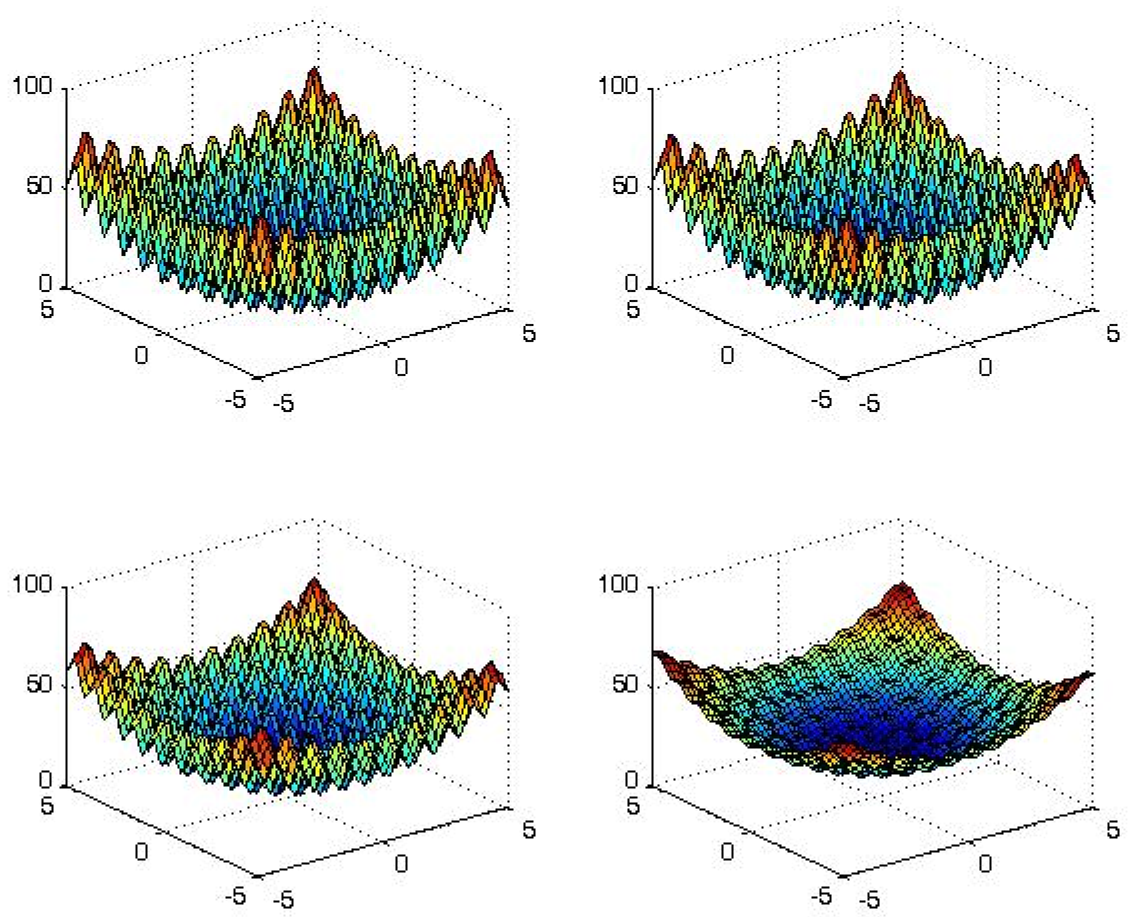


Figure 1.3: Rastrigin Smoothing

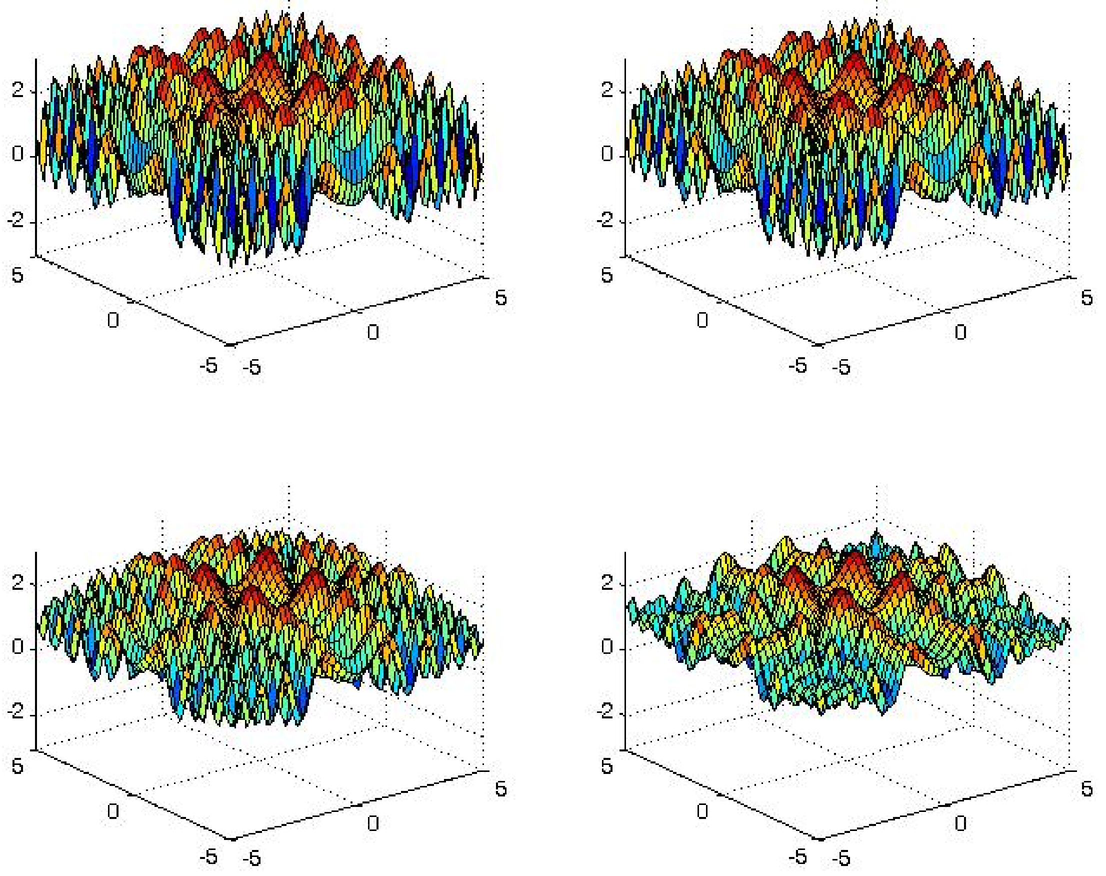


Figure 1.4: Smoothing for $f(x,y) = 1 + \sin x^2 + \sin y^2 - 0.1e^{-x^2-y^2}$ (Example from CGO)

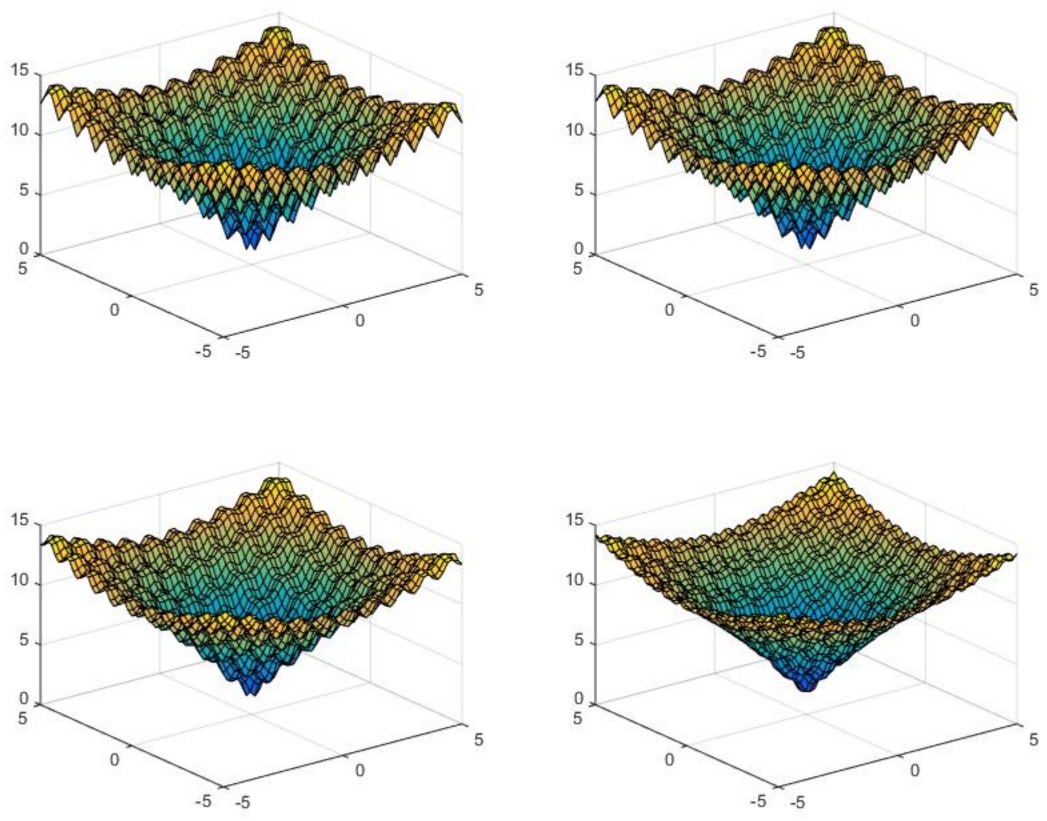


Figure 1.5: Ackley Smoothing

1.4 Original Smoothing Algorithm and Our Modified Smoothing Techniques

In CGO, Simulated Annealing with Smoothing Algorithm is:

Algorithm 1 Simulated Annealing with Smoothing in CGO

```
Given $Ts, L, Kb, \Delta$ sequence:$Ds$, $x_0$
for $k = 1, ...length(Ts)$ do
  $L$ = number of moves to attempt, $T = Ts(k)$, $\Delta = Ds(k)$.
  Search region $sr = (-lb + ub) * \frac{T}{\max(Ts)}$
  for $m = 1$ to $L$ do
    Randomly generate a new neighbouring solution $x_{new}$ in $[x_{old} - sr, x_{old} + sr]$ and
    make sure that $x_{new}$ does not exceed the boundaries.
    Evaluate $\bar{f}_{new} = f(x_{new}) + \frac{1}{6}\Delta^2 trace(H)$.
    if $\bar{f}_{new} < \bar{f}_{old}$ then
      Accept this new solution, and update the solution.
    else
      Accept with probability $P(T) = e^{\frac{-(\bar{f}_{new} - \bar{f}_{old})}{Kb \cdot T}}$.
      Update the solution if accepted.
    end if
  end for
end for
```

We did two modifications for this algorithm:

- First, instead of letting temperature sequence and Δ sequence have the same length, we let the Δ sequence contains only several entries depends on the problem. For each entry in the Δ sequence, we run through all the temperature sequence and use the result for the next entry of Δ sequence.
- Second, we add a *srmin* parameter to control the search region does not become too small. Cause if we use $Ts = T_0 * 0.9^{0:199}$ for example, $0.9^{199} = 7.8 \times 10^{-10}$, $0.9^{99} = 2.95 \times 10^{-5}$ so basically after running 99 entries in Ts, the search region is less than $(-lb + ub) * 2.95 \times 10^{-5}$ so the point almost stay the same for the last 100 entries of Ts.

So our new Simulated Annealing with Smoothing Technique Algorithm is:

Algorithm 2 Modified Simulated Annealing with $Trace - Smooth$

```
Given $Ts, L, Kb, \Delta$ sequence:$Ds$, $x_0, srmin$
for $i = 1, ...length(Ds)$ do
  $\Delta = Ds(i)$
  for $k = 1, ...length(Ts)$ do
    $L$ = number of moves to attempt, $T = Ts(k)$.
    Search region $sr = \max\{(-lb + ub) * \frac{T}{\max(Ts)}, srmin\}$
    for $m = 1$ to $L$ do
      Randomly generate a new neighbouring solution $x_{new}$ in $[x_{old} - sr, x_{old} + sr]$ and
      make sure that $x_{new}$ does not exceed the boundaries.
      Evaluate $\bar{f}_{new} = f(x_{new}) + \frac{1}{6}\Delta^2 trace(H)$.
      if $\bar{f}_{new} < \bar{f}_{old}$ then
        Accept this new solution, and update the solution.
      else
        Accept with probability $P(T) = e^{\frac{-(\bar{f}_{new} - \bar{f}_{old})}{Kb \cdot T}}$.
        Update the solution if accepted.
      end if
    end for
  end for
end for
```

The numerical results show that our modifications made the simulated annealing more efficient and can do better than before.

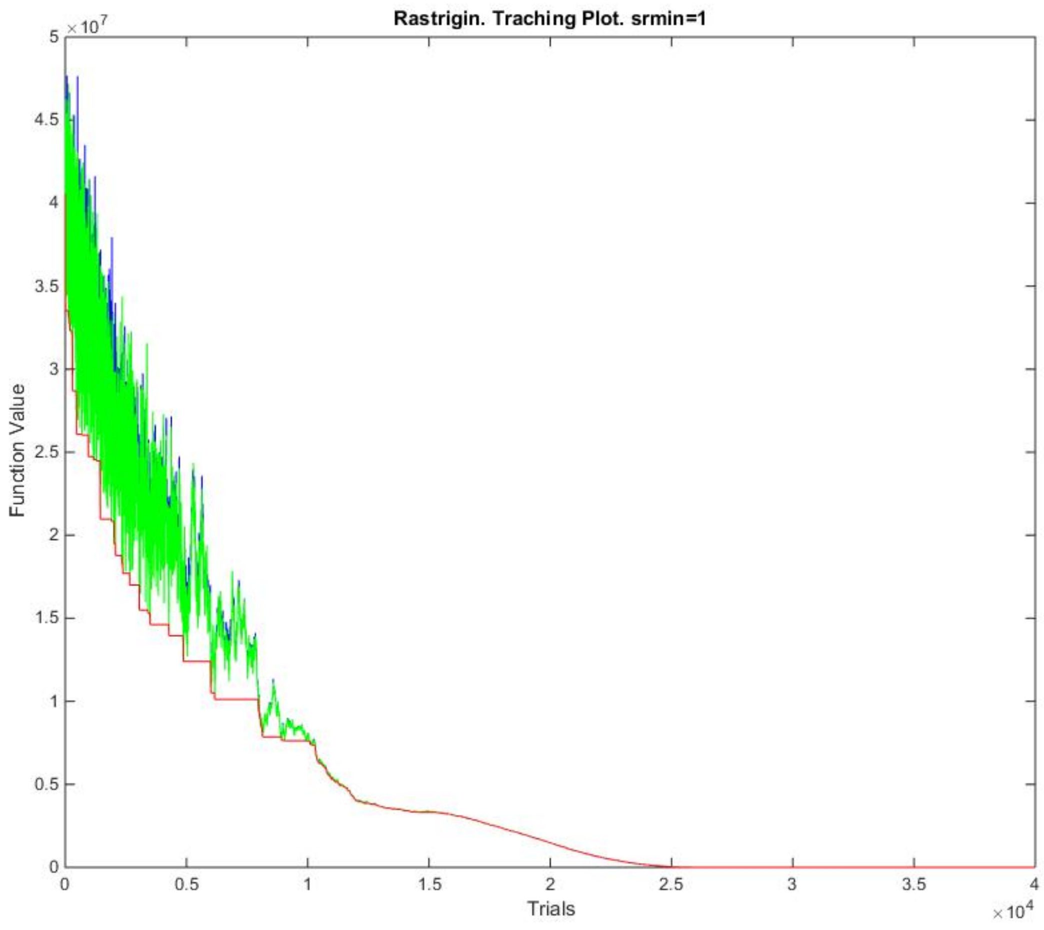


Figure 1.6: Rastrigin. Tracking Plot. srmin=1.

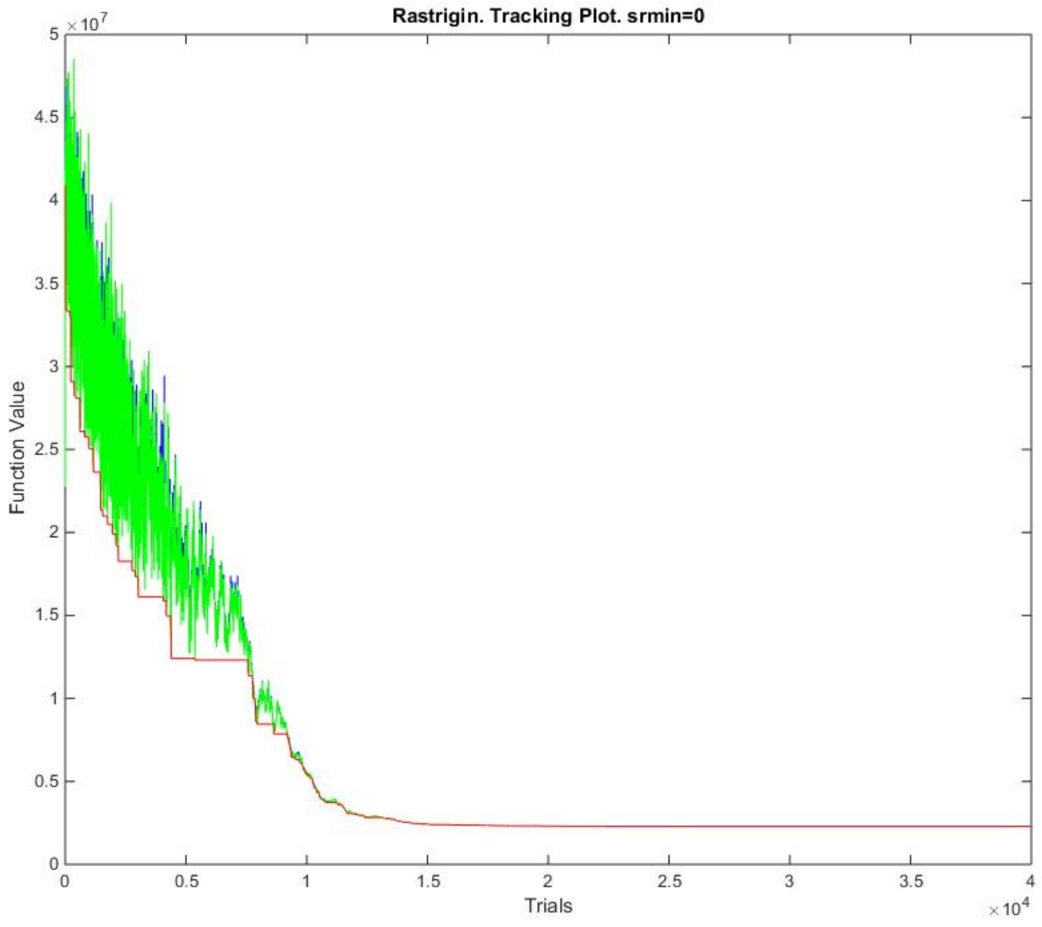


Figure 1.7: Rastrigin. Tracking Plot. srmin=0 (original).

srmin should be defined according to the search range of the problem. Also you can set it to 0 to use the original search range in CGO.

Similarly, the algorithm of simulated annealing with $\lambda - Smooth$ is:

Algorithm 3 Modified Simulated Annealing with $\lambda - Smooth$

```
Given $Ts, L, Kb, \lambda$ sequence:$Ls$, $x_0, x_*, srmin$
for $i = 1, ...length(Ds)$ do
  $\lambda = Ls(i)$
  for $k = 1, ...length(Ts)$ do
    $L$ = number of moves to attempt, $T = Ts(k)$.
    Search region $sr = \max\{(-lb + ub) * \frac{T}{\max(Ts)}, srmin\}$
    for $m = 1$ to $L$ do
      Randomly generate a new neighbouring solution $x_{new}$ in $[x_{old} - sr, x_{old} + sr]$ and
      make sure that $x_{new}$ does not exceed the boundaries.
      Evaluate $\bar{f}_{new} = f(x_{new}) + \lambda||x - x_*||_2^2$.
      if $\bar{f}_{new} < \bar{f}_{old}$ then
        Accept this new solution, and update the solution.
      else
        Accept with probability $P(T) = e^{\frac{-(\bar{f}_{new} - \bar{f}_{old})}{Kb \cdot T}}$.
        Update the solution if accepted.
      end if
    end for
  end for
end for
```

Notice that simulated annealing is phase 1. For phase 2 we use $fmincon$ with start point(s) came from phase 1 to get the final results.

Chapter 2

Numerical Results

We use three problems which have many local optima and are very difficult to find the global optimum. They are:

- Griewank
- Rastrigin
- Ackley

And test the four methods:

- Simulated Annealing with $Trace - Smooth$
- Simulated Annealing with $\lambda - Smooth$
- Pure Simulated Annealing
- MATLAB build-in function $fmincon$ (Trust-Region)

We set up the same parameters for the 3 SA methods and use same start point for 4 methods.

2.1 Griewank

Griewank function is:

$$f(x) = p * \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{2.1}$$

Usually $p = \frac{1}{4000}$ and we can control p to control the shape of the function. It has many local optima and the global one is $x_* = [0, 0, ..., 0]^T$, $f(x_*) = 0$.

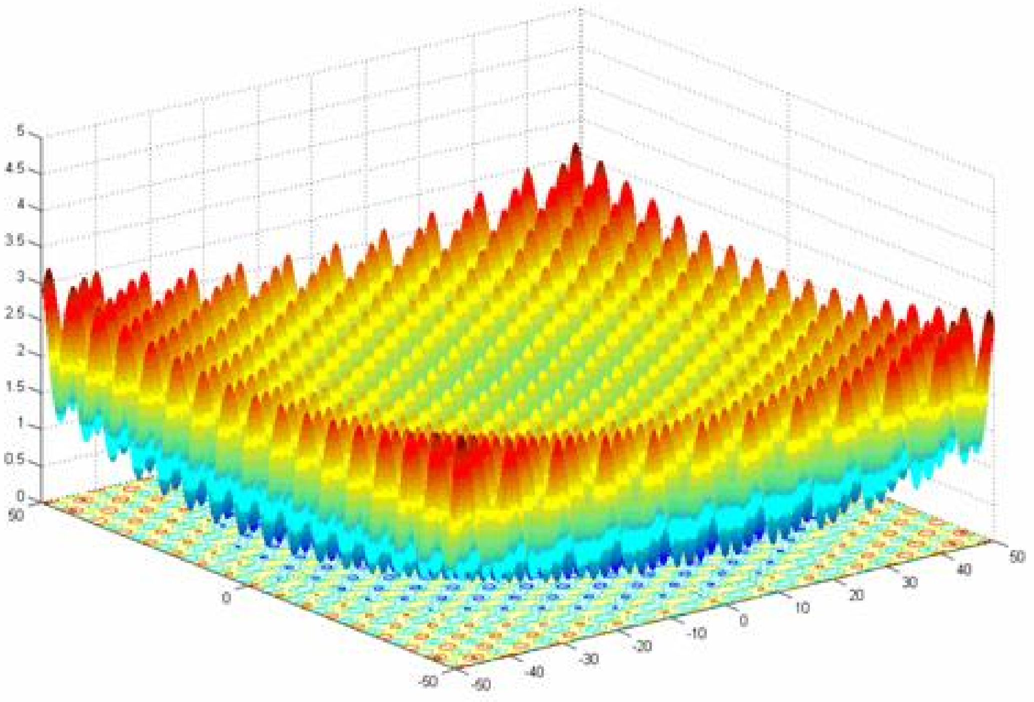
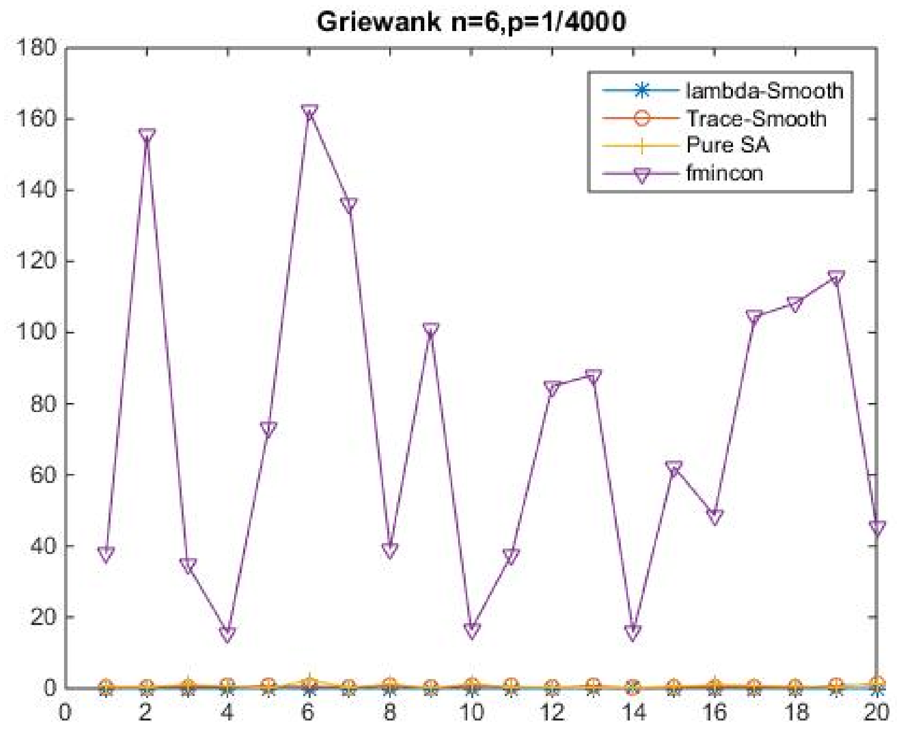


Figure 2.1: Griewank

We use a randomly generated x_* from $[-1, -1, ..., -1]^T$ to $[1, 1, ..., 1]^T$. Although we know the exact value of x_* but we pretend we don't know to be fair for other methods. The search region is from $[-1000, -1000, ..., -1000]^T$ to $[1000, 1000, ..., 1000]^T$.

We did 20 cases using different start point for $n = 6$ and $p = \frac{1}{4000}$ of Griewank. $x - axis$ is the number of cases and $y - axis$ is the final function value.



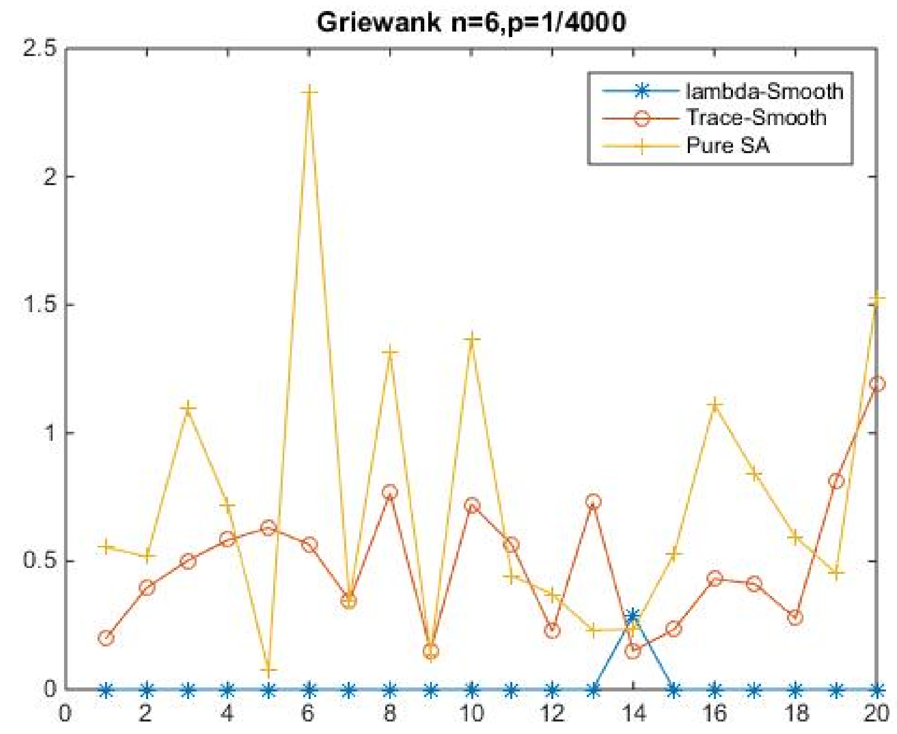


Figure 2.2: Compare SA Methods with $fmincon$

The first figure shows that the three simulated annealing methods are better than just use $fmincon$ cause Griewank function is very hard for $fmincon$(Trust-Region) to solve.

The second figure we remove the $fmincon$ line to compare our three simulated annealing methods. We can see that our $\lambda - Smooth$ is doing best and $Trace - Smooth$ is better than pure simulated annealing in most cases(16 out of 20).

As we mentioned before, we can control the value of p to change the shape of Griewank.

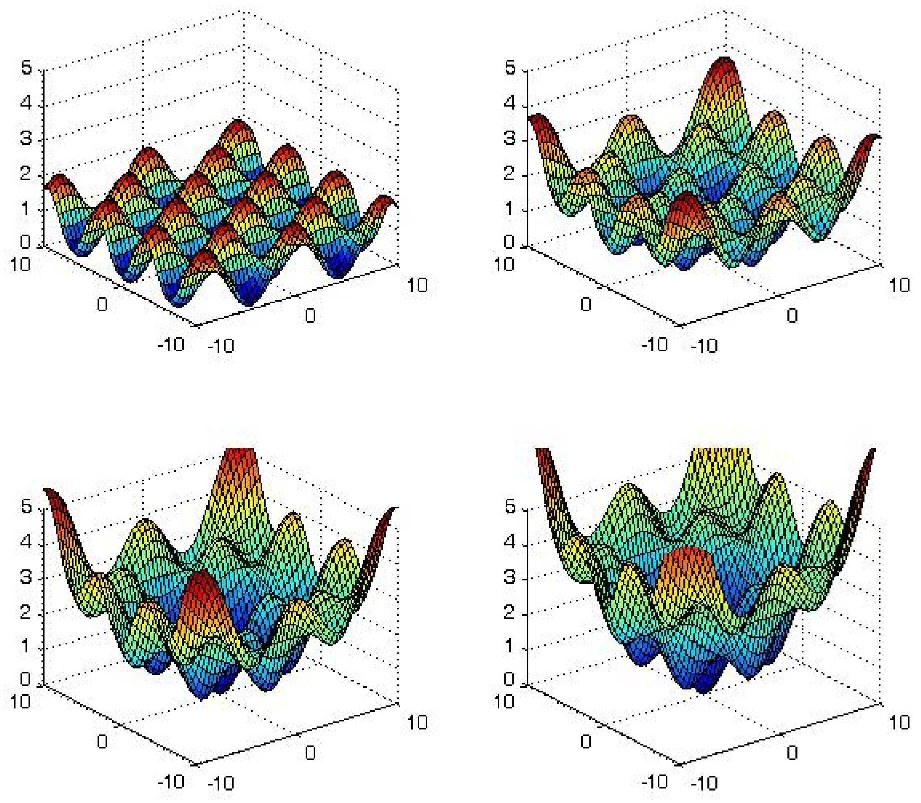


Figure 2.3: Shape of Griewank if p becomes greater

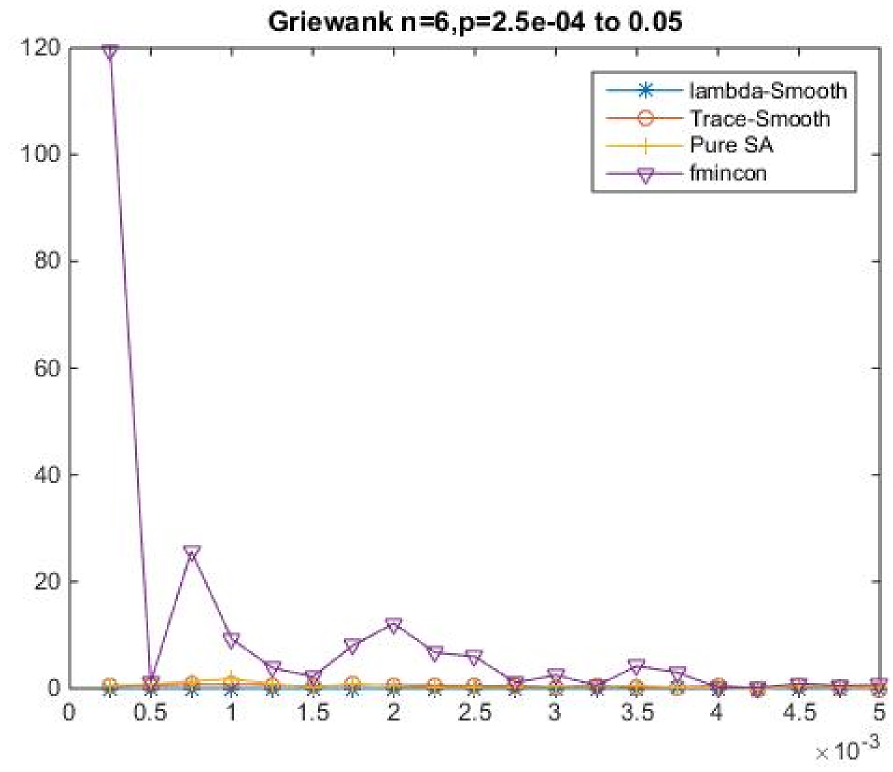


Figure 2.4: Compare SA methods with $fmincon$ when function becomes flattened

$x - axis$ is the value of p and $y - axis$ is the final function value.

As p goes greater, the function shape is more flattened. So $fmincon$ can do better because the function is not so nasty. In this situation the results of $fmincon$ and our 3 SA methods doesn't show much difference.

2.2 Rastrigin

Rastrigin function is:

$$f(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)) \tag{2.2}$$

The global optima is $x_* = [0, 0, ..., 0]^T$ and $f(x_*) = 0$.

As n goes bigger, it is very difficult to find the global optima using simulated annealing or trust-region.

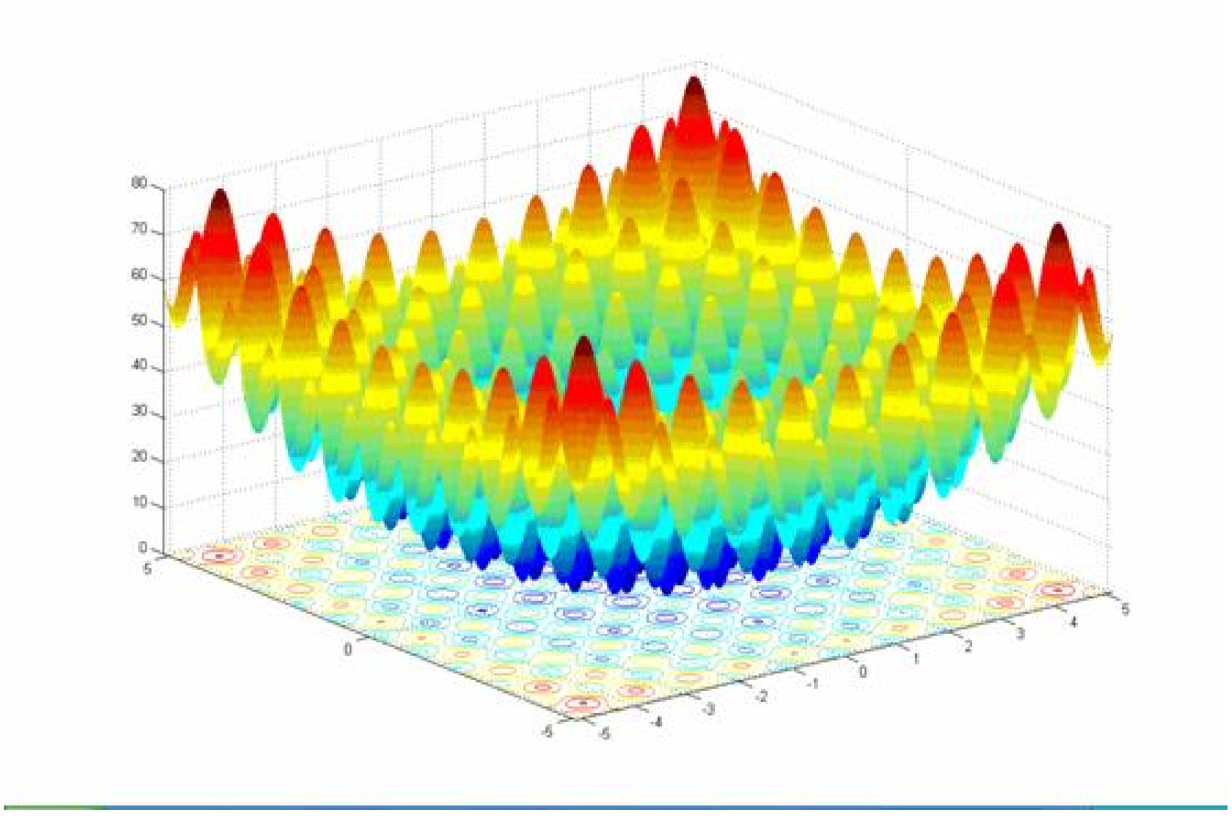


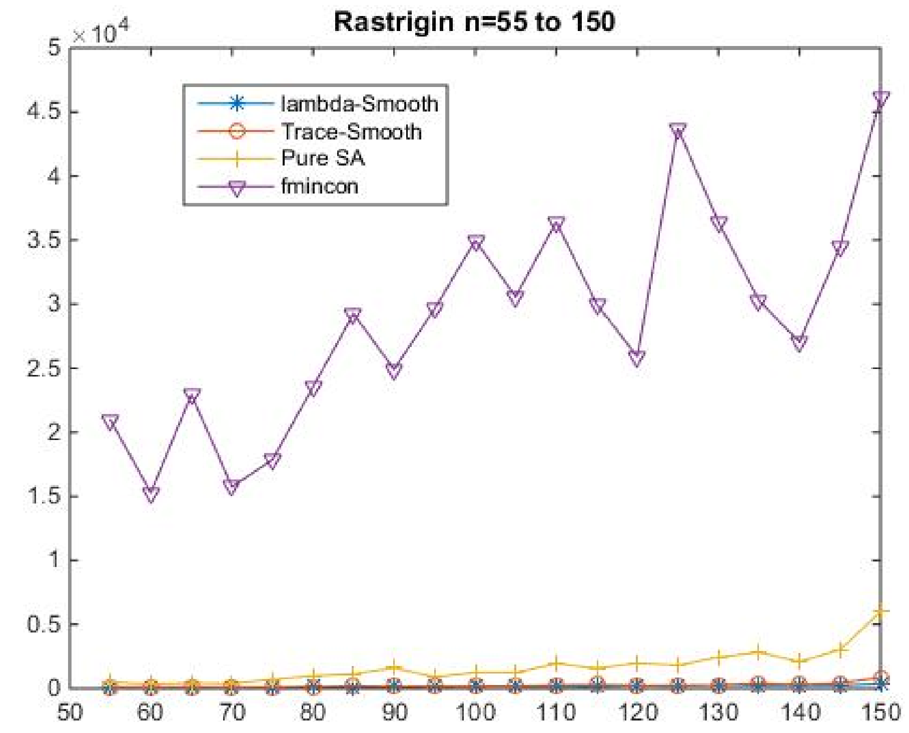
Figure 2.5: Rastrigin

In this example we use n from 55 to 150 to see the difference of each method. Same as we did in Griewank example, we use a randomly generated x_* from $[-1, -1, ..., -1]^T$ to $[1, 1, ..., 1]^T$. The search region is from $[-1000, -1000, ..., -1000]^T$ to $[1000, 1000, ..., 1000]^T$.

$x - axis$ is dimension n and $y - axis$ is the final function value.

First two figures are the result for the comparison of four methods.

The last two figures are the results and time we compare our three simulated annealing methods with MATLAB build-in SA function: *simulannealbnd*.



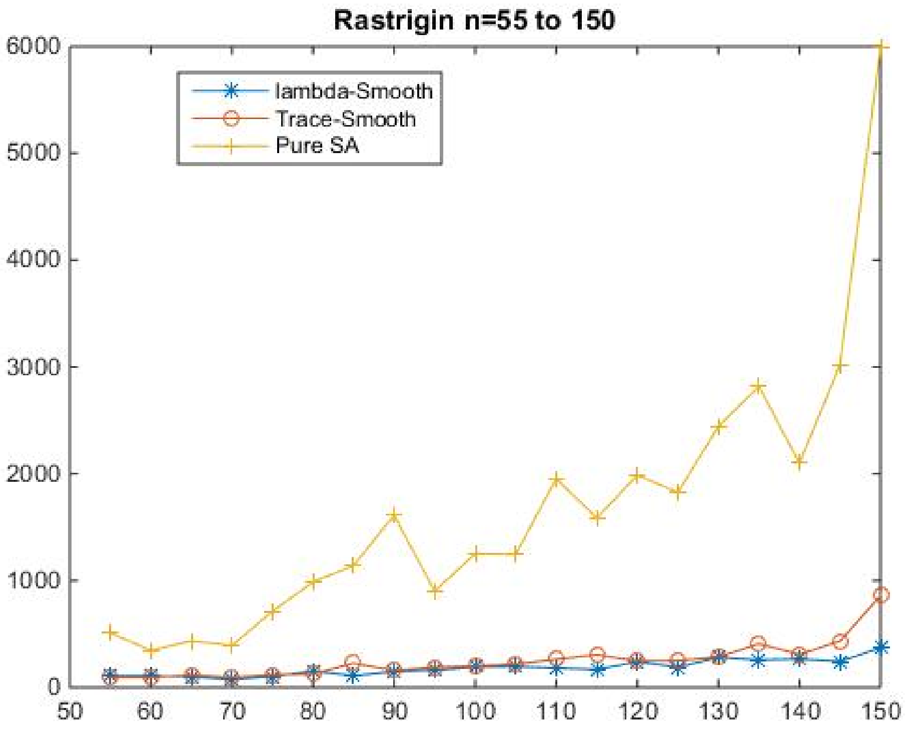
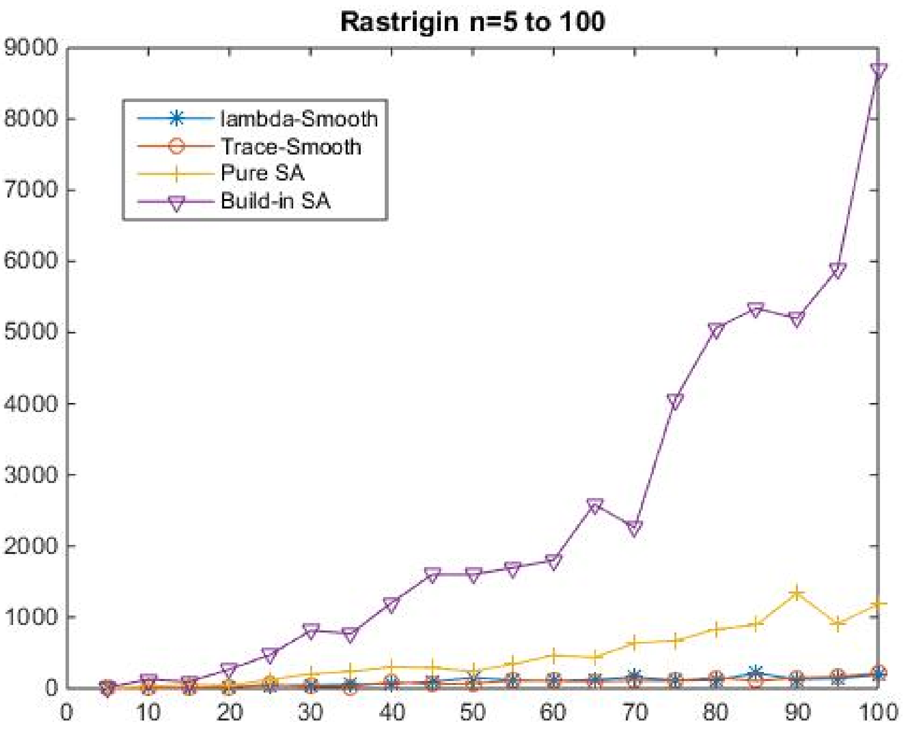


Figure 2.6: Compare SA Methods with $fmincon$



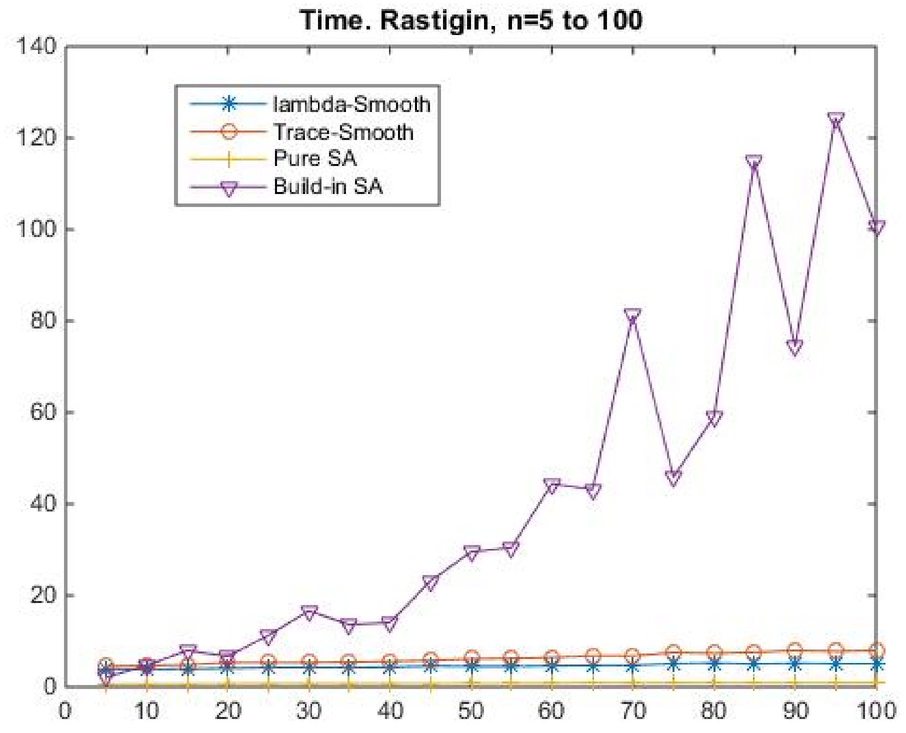


Figure 2.7: Compare Our 3 SA Methods with MATLAB Build-in SA function

From the first two figures we can see clearly that our smoothing techniques really work cause they can get a much better results than just using the pure simulated annealing.

From the last two figures we can see that our simulated annealing code is much more efficient than MATLAB build-in function. We take less time and get better answer. The reason probably is that *simulannealbnd* only input the function handle, x_0, upper and lower bound but no other parameters.

2.3 Ackley

The Ackley function is:

$$f(x) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} \tag{2.3}$$

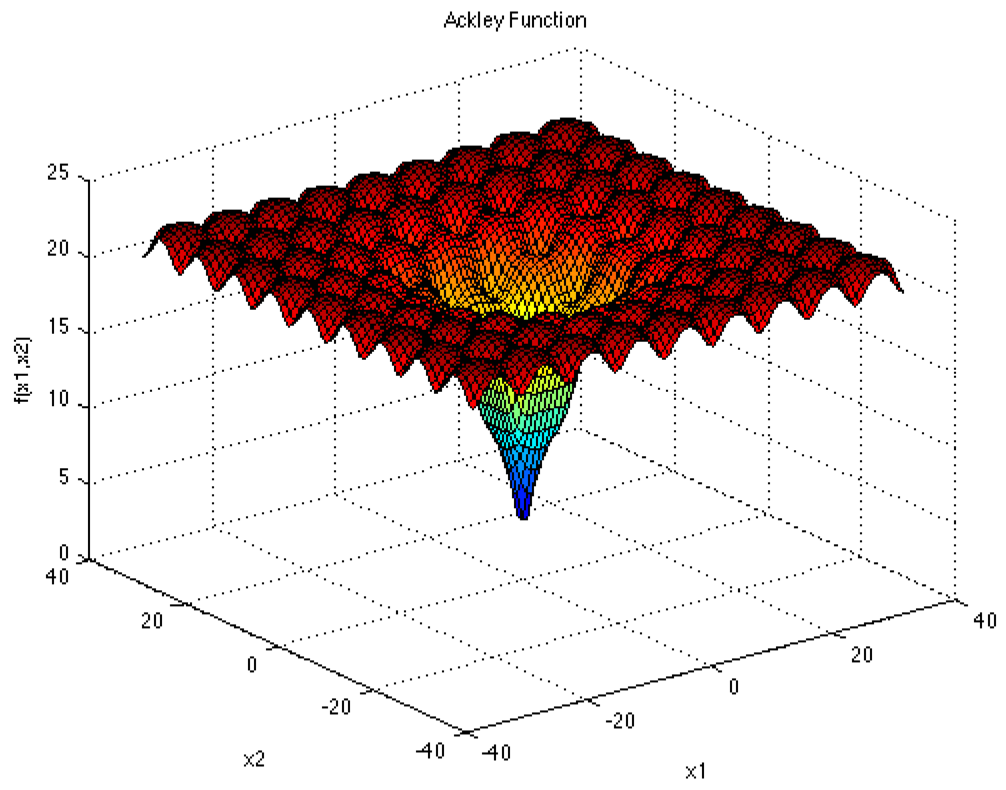


Figure 2.8: Ackley function for $n = 2$

There are many local optima and one global optimum at $x_* = [0, 0, ..., 0]^T$. $f(x_*) = 0$. However, as n goes bigger, it is very difficult to find the global optimum. The search region is: $[-30, -30, ..., -30]^T$ to $[30, 30, ..., 30]^T$.

$x - axis$ is the dimension n and $y - axis$ is the final function value.

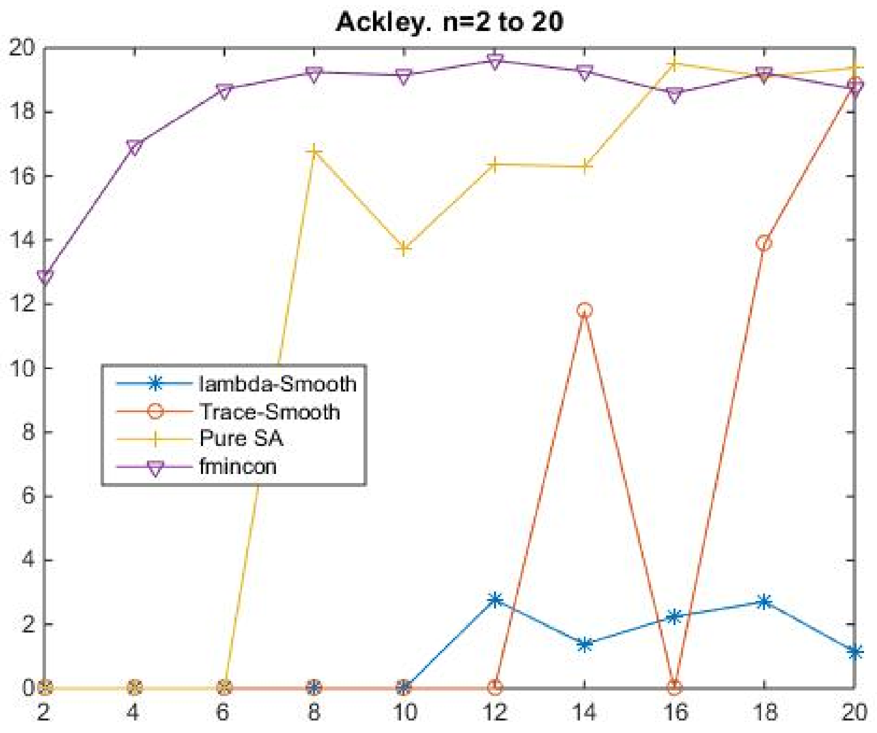


Figure 2.9: Ackley example. $n = 2$ to 20

For this example, first we can say that $\lambda - Smooth$ doing well as usual. However, $Trace - Smooth$ is not so stable and we think that is probably because we didn't set the Δ-sequence right. For the previous examples, we did lots of experiments to find the suitable Δ-sequence for each problem. But this time we use $ADMAT$ to compute H and it takes quite a long time to run the $Trace - Smooth$ simulated annealing(Took 3.5 hours to get this figure). We believe that $Trace - Smooth$ technique can do better if we set up the right Δ-sequence since the picture we showed in figure 1.5 illustrates that $Trace - Smooth$ technique can smooth the Ackley function very well.

Chapter 3

Trust-Region Method with Simulated Annealing and Smoothing Technique

3.1 Introduction

We have tested the Trust-Region Method combined with Simulated Annealing and found it is an improvement over just using Trust-Region Method (see another report for detail). Now we want to combine Trust-Region Method with Simulated Annealing and Smoothing Technique.

For Trust-Region Method, we need to compute gradient g and Hessian matrix H to solve the Trust-Region sub-problem for each iteration.

For $\lambda - Smooth$, $\bar{f}(x) = f(x) + \lambda ||x - x_*||_2^2$, since $||x - x_*||_2^2 = \sum_{i=1}^{n}(x_i - x_{*i})^2$. So we have:

$$\bar{g} = g + 2\lambda(x - x_*) \tag{3.1}$$

$$\bar{H} = H + 2\lambda I \tag{3.2}$$

Where g and H is the gradient and Hessian matrix of original function $f(x)$.

For $Trace - Smooth$ $\bar{f}(x) = f(x) + \frac{1}{6}\Delta^2 trace(H)$, since $trace(H) = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$. So we have:

$$\bar{g} = g + \frac{1}{6}\Delta^2 \sum_{k=1}^{n} \frac{\partial^3 f}{\partial x_k^2 \partial x_i} \tag{3.3}$$

$$\bar{H} = H + \frac{1}{6}\Delta^2 R \tag{3.4}$$

Where g and H is the gradient and Hessian matrix of original function $f(x)$ and matrix R is:

$$R_{ij} = \sum_{k=1}^{n} \frac{\partial^4 f}{\partial x_k^2 \partial x_i \partial x_j} \tag{3.5}$$

Since usually we can not get the information of the third and forth derivation, so we decide just use the original function $f(x)$'s gradient and Hessian matrix for each iteration and use the formula $\bar{f}(x) = f(x) + \frac{1}{6}\Delta^2 trace(H)$ to compute the function's value. Now we give the four algorithms for Trust-Region method with simulated annealing and smoothing techniques.

Algorithm 4 Trust Region Method (TRM)

```
Given $\tau_1, \tau_2, \gamma 1, \gamma 2$
while $norm(g) > tol$ and number of iterations$<$ $itbnd$ do
    Solve TRS for $s_k$, assign $qpval_k \leftarrow \nabla f_k^T s_k + \frac{1}{2} s_k^T \nabla^2 f_k s_k$
    $newval_k \leftarrow f(x_k + s_k)$
    $ratio_k \leftarrow \frac{newval_k - f_k}{qpval_k}$
    Adjust $\Delta_k$
    if $ratio_k < \tau_1$ then
        $\Delta_{k+1} \leftarrow \frac{||s_k||}{\gamma_1}$
    else
        if $ratio_k > \tau_2$,and $||s_k|| = \Delta_k$ then
            $\Delta_{k+1} \leftarrow \gamma_2 \Delta_k$
        else
            $\Delta_{k+1} \leftarrow \Delta_k$
        end if
    end if
    Update $x$:
    if $ratio_k \leq 0$ then
        $x_{k+1} \leftarrow x_k$
    else
        $x_{k+1} \leftarrow x_k + s_k$
    end if
end while
```

Algorithm 5 Trust-Region Method with Simulated Annealing

```
Given temperature sequence $T$, $Trials$, $x_0$, $m = 0$
*Phase 1:*
**for** $n = 1, ..., length(T)$ **do**
   Given $\tau_1, \tau_2, \gamma 1, \gamma 2$, $t = T(n)$, $x = x_0$
   **while** $norm(g) > tol$ and number of iterations$<$ $Trials$ **do**
      Solve TRS for $s_k$, assign $qpval_k \leftarrow \nabla f_k^T s_k + \frac{1}{2} s_k^T \nabla^2 f_k s_k$
      $newval_k \leftarrow f(x_k + s_k)$
      $ratio_k \leftarrow \frac{newval_k - f_k}{qpval_k}$
      Adjust $\Delta_k$
      **if** $ratio_k < \tau_1$ **then**
         $\Delta_{k+1} \leftarrow \frac{||s_k||}{\gamma_1}$
      **else**
         **if** $ratio_k > \tau_2$,and $||s_k|| = \Delta_k$ **then**
            $\Delta_{k+1} \leftarrow \gamma_2 \Delta_k$
         **else**
            $\Delta_{k+1} \leftarrow \Delta_k$
         **end if**
      **end if**
      Update $x$:
      **if** $ratio_k \geq 0$ or $exp(\frac{ratio_k}{t}) > rand$ **then**
         $x_{k+1} \leftarrow x_k + s_k$
      **else**
         $x_{k+1} \leftarrow x_k$
      **end if**
   **end while**
   $f_n = f(x_k)$
   **if** $f_n$ is different from any $f_1$ to $f_{n-1}$ **then**
      $m = m + 1$, $\overline{x_m} = x_k$, ;
   **end if**
**end for**

*Phase 2:*
**for** $k = 1, ..., m$ **do**
   Algorithm 4 with initial point $\overline{x_k}$
**end for**
Output the smallest $f$ in phase 2 and its point.
```

Algorithm 6 Trust-Region Method with Simulated Annealing and $\lambda - Smooth$

```
Given temperature sequence $T$, $Trials$, $x_0$, $m = 0$, $x_*$, $\lambda$ sequence:$Ls$.
*Phase 1:*
for $l = 1, ..., length(Ls)$ do
   for $n = 1, ..., length(T)$ do
      Given $\tau_1, \tau_2, \gamma 1, \gamma 2$, $t = T(n)$, $\lambda = Ls(l)$, $x = x_0$
      while $norm(\bar{g}) > tol$ and number of iterations$<$ $Trials$ do
         Solve TRS for $s_k$ with $\bar{g} = g + 2\lambda(x - x_*)$ and $\bar{H} = H + 2\lambda I$, assign $qpval_k \leftarrow \nabla f_k^T s_k + \frac{1}{2} s_k^T \nabla^2 f_k s_k$
         $newval_k \leftarrow f(x_k + s_k) + \lambda ||x_k + s_k - x_*||_2^2$
         $ratio_k \leftarrow \frac{newval_k - (f_k + \lambda ||x_k - x_*||_2^2)}{qpval_k}$
         Adjust $\Delta_k$
         if $ratio_k < \tau_1$ then
            $\Delta_{k+1} \leftarrow \frac{||s_k||}{\gamma_1}$
         else
            if $ratio_k > \tau_2$,and $||s_k|| = \Delta_k$ then
               $\Delta_{k+1} \leftarrow \gamma_2 \Delta_k$
            else
               $\Delta_{k+1} \leftarrow \Delta_k$
            end if
         end if
         Update $x$:
         if $ratio_k \geq 0$ or $exp(\frac{ratio_k}{t}) > rand$ then
            $x_{k+1} \leftarrow x_k + s_k$
         else
            $x_{k+1} \leftarrow x_k$
         end if
      end while
      $f_n = f(x_k)$
      if $f_n$ is different from any $f_1$ to $f_{n-1}$ then
         $m = m + 1$, $\overline{x_m} = x_k$, ;
      end if
   end for
end for

*Phase 2:*
for $k = 1, ..., m$ do
   Algorithm 4 with initial point $\overline{x_k}$
end for
Output the smallest $f$ in phase 2 and its point.
```

Algorithm 7 Trust-Region Method with Simulated Annealing and $Trace - Smooth$

```
Given temperature sequence $T$, $Trials$, $x_0$, $m = 0$, $x_*$, $\Delta$ sequence: $Ds$.
Phase 1:
for $l = 1, ..., length(Ds)$ do
  for $n = 1, ..., length(T)$ do
    Given $\tau_1, \tau_2, \gamma 1, \gamma 2$, $t = T(n)$, $\delta = Ds(l)$, $x = x_0$
    while $norm(g) > tol$ and number of iterations$< Trials$ do
      Solve TRS for $s_k$, assign $qpval_k \leftarrow \nabla f_k^T s_k + \frac{1}{2} s_k^T \nabla^2 f_k s_k$
      $newval_k \leftarrow f(x_k + s_k) + \frac{1}{6}\delta^2 Trace(H(x_k + s_k))$
      $ratio_k \leftarrow \frac{newval_k - (f_k + \frac{1}{6}\delta^2 Trace(H(x_k)))}{qpval_k}$
      Adjust $\Delta_k$
      if $ratio_k < \tau_1$ then
        $\Delta_{k+1} \leftarrow \frac{||s_k||}{\gamma_1}$
      else
        if $ratio_k > \tau_2$,and $||s_k|| = \Delta_k$ then
          $\Delta_{k+1} \leftarrow \gamma_2 \Delta_k$
        else
          $\Delta_{k+1} \leftarrow \Delta_k$
        end if
      end if
      Update $x$:
      if $ratio_k \geq 0$ or $exp(\frac{ratio_k}{t}) > rand$ then
        $x_{k+1} \leftarrow x_k + s_k$
      else
        $x_{k+1} \leftarrow x_k$
      end if
    end while
    $f_n = f(x_k)$
    if $f_n$ is different from any $f_1$ to $f_{n-1}$ then
      $m = m + 1$, $\overline{x_m} = x_k$, ;
    end if
  end for
end for

Phase 2:
for $k = 1, ..., m$ do
  Algorithm 4 with initial point $\overline{x_k}$
end for
Output the smallest $f$ in phase 2 and its point.
```

3.2 Numerical Results

Now we compare our 7 methods to see the difference and get some conclusions. Our 7 methods are:

- Pure Simulated Annealing Method
- Simulated Annealing with $\lambda - Smooth$
- Simulated Annealing with $Trace - Smooth$
- Traditional Trust-Region Method
- Trust-Region Method with Simulated Annealing
- Trust-Region Method with Simulated Annealing and $\lambda - Smooth$
- Trust-Region Method with Simulated Annealing and $Trace - Smooth$

First we use Rastrigin function to test these 7 methods. The results are shown below:

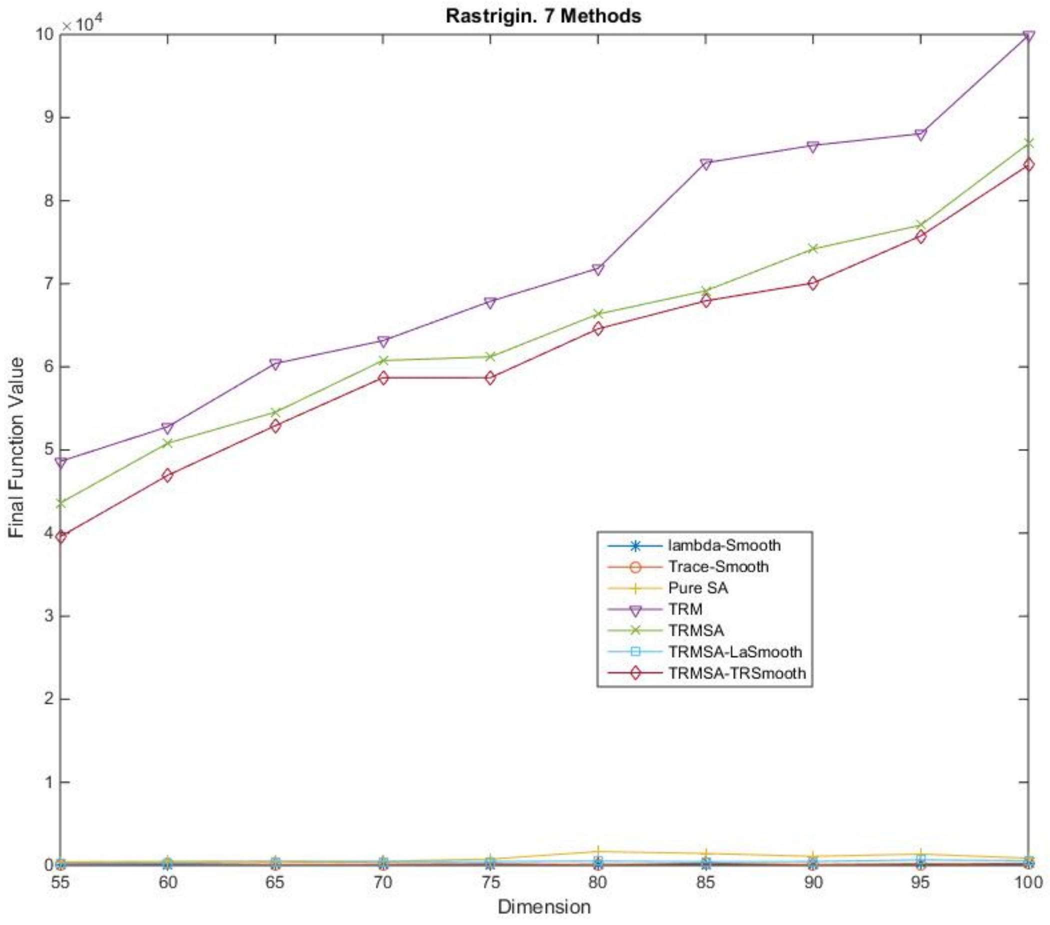


Figure 3.1: Rastrigin example. 7 Methods. $n = 55$ to 100

As we can see, traditional TRM, TRMSA and TRMSA with $Trace - Smooth$ do not give us a satisfied results. Probably because Rastrigin has many local optima and it is very difficult for TRM to solve. Also, since we just use the original function $f(x)$'s gradient and Hessian matrix so the TRMSA with $Trace - Smooth$ does not do well either.

Below we show the plot of four methods which did quite well for this problem (the ones of bottom lines): Pure SA, SA with $Trace - Smooth$, SA with $\lambda - Smooth$ and TRMSA with $\lambda - Smooth$.

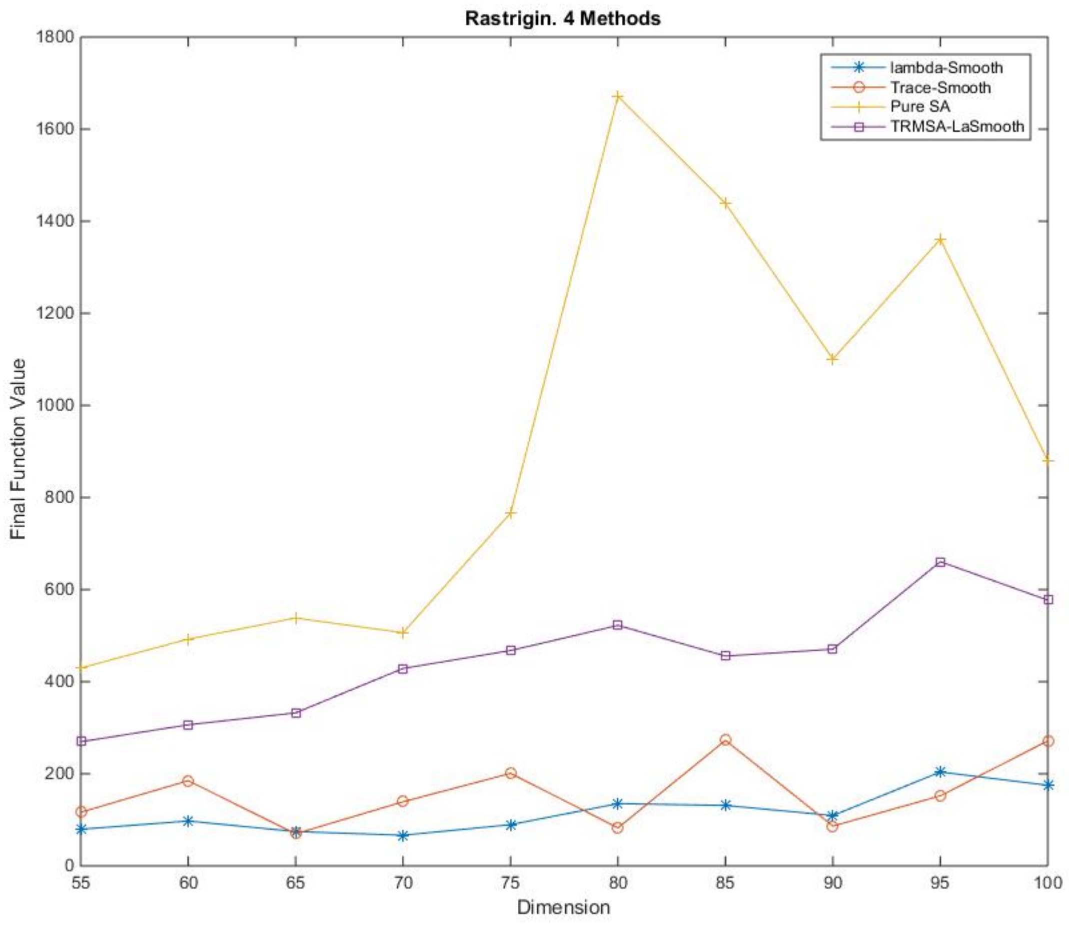


Figure 3.2: Rastrigin example. 4 Methods. $n = 55$ to 100

Now we can give a conclusion which is: for those problems that have many local optima, TRM, TRMSA and TRMSA with $Trace - Smooth$ probably can not do well. For the rest 4 methods that can do quite well in those problems, Pure SA may be the worst, and SA with $Trace - Smooth$ and SA with $\lambda - Smooth$ can do best. TRMSA with $\lambda - Smooth$ is between them.

Now we show the results of solving the optimization problem for Griewank and modified Griewank function. In this case, the problem is not so difficult to solve, so the difference between our 7 methods is not so big but we can see the conclusions we drew before still hold.

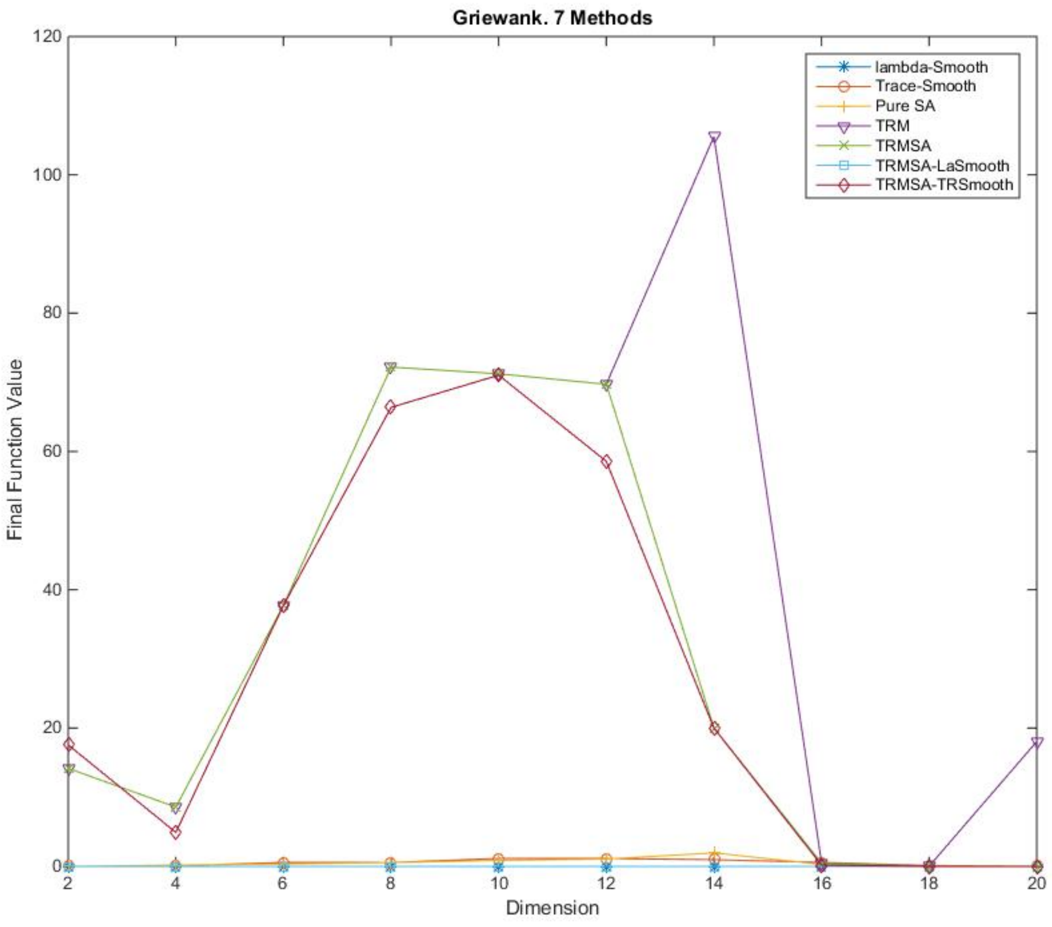


Figure 3.3: Griewank example. 7 Methods. $n = 2$ to 20

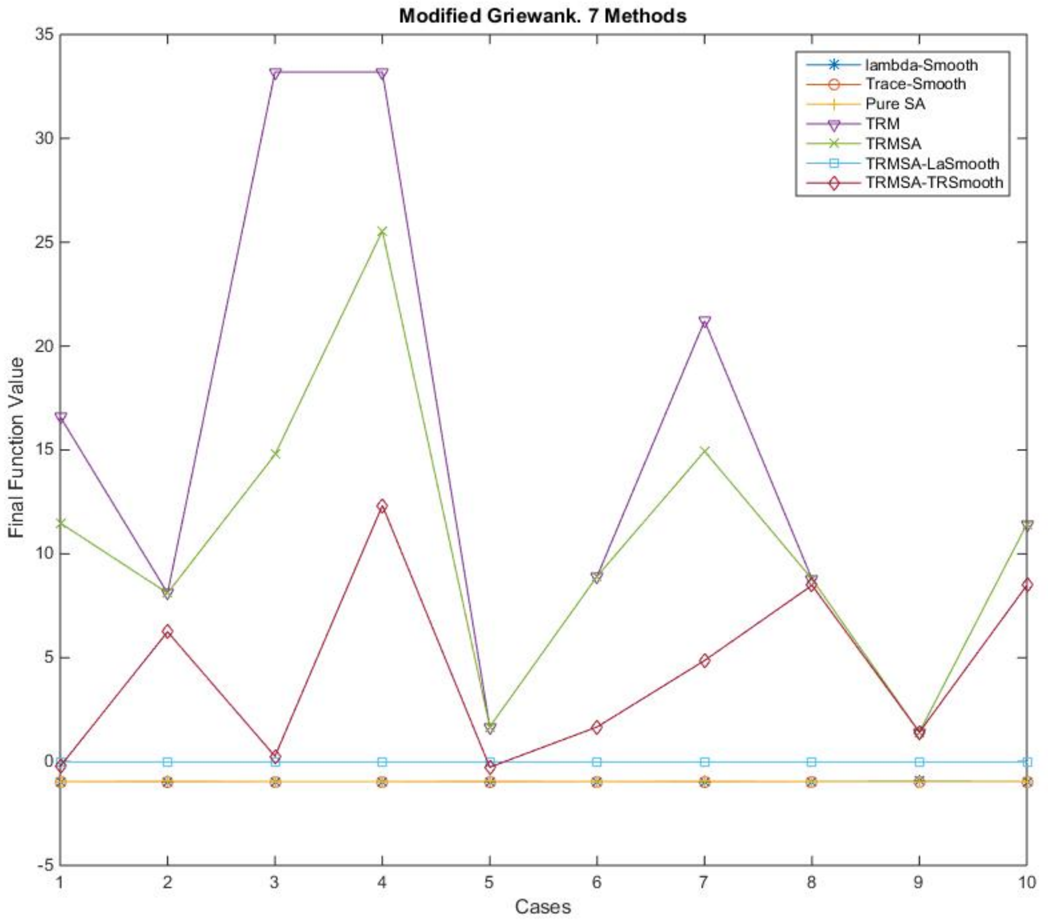


Figure 3.4: Modified Griewank example. 7 Methods. $n = 2$. 10 Cases

After doing these experiments, we move forward to focus on 4 methods:

- SA with $Trace - Smooth$
- SA with $\lambda - Smooth$
- TRMSA with $\lambda - Smooth$
- TRM with $\lambda - Smooth$

The last one is a new method cause we find it takes much fewer evaluations of $f(x)$ and Hessian matrix but can still get a good result. And the algorithm is :

Algorithm 8 Trust-Region Method with $\lambda - Smooth$

```
Given $x_0$, $m = 0$, $x_*$, $\lambda$ sequence:$Ls$.
*Phase 1:*
**for** $l = 1, ..., length(Ls)$ **do**
    Given $\tau_1, \tau_2, \gamma 1, \gamma 2$, $\lambda = Ls(l)$
    **while** $norm(g) > tol$ and number of iterations< $itbnd$ **do**
        Solve TRS for $s_k$ with $\bar{g} = g + 2\lambda(x - x_*)$ and $\bar{H} = H + 2\lambda I$, assign $qpval_k \leftarrow \nabla f_k^T s_k + \frac{1}{2} s_k^T \nabla^2 f_k s_k$
        $newval_k \leftarrow f(x_k + s_k) + \lambda||x_k + s_k - x_*||_2^2$
        $ratio_k \leftarrow \frac{newval_k - (f_k + \lambda||x_k - x_*||_2^2)}{qpval_k}$
        Adjust $\Delta_k$
        **if** $ratio_k < \tau_1$ **then**
            $\Delta_{k+1} \leftarrow \frac{||s_k||}{\gamma_1}$
        **else**
            **if** $ratio_k > \tau_2$,and $||s_k|| = \Delta_k$ **then**
                $\Delta_{k+1} \leftarrow \gamma_2 \Delta_k$
            **else**
                $\Delta_{k+1} \leftarrow \Delta_k$
            **end if**
        **end if**
        Update $x$:
        **if** $ratio_k \geq 0$ **then**
            $x_{k+1} \leftarrow x_k + s_k$
        **else**
            $x_{k+1} \leftarrow x_k$
        **end if**
    **end while**
    $f_n = f(x_k)$
    **if** $f_n$ is different from any $f_1$ to $f_{n-1}$ **then**
        $m = m + 1$, $\overline{x_m} = x_k$, ;
    **end if**
**end for**

*Phase 2:*
**for** $k = 1, ..., m$ **do**
    Algorithm 4 with initial point $\overline{x_k}$
**end for**
Output the smallest $f$ in phase 2 and its point.
```

Now we give some results about these 4 methods.

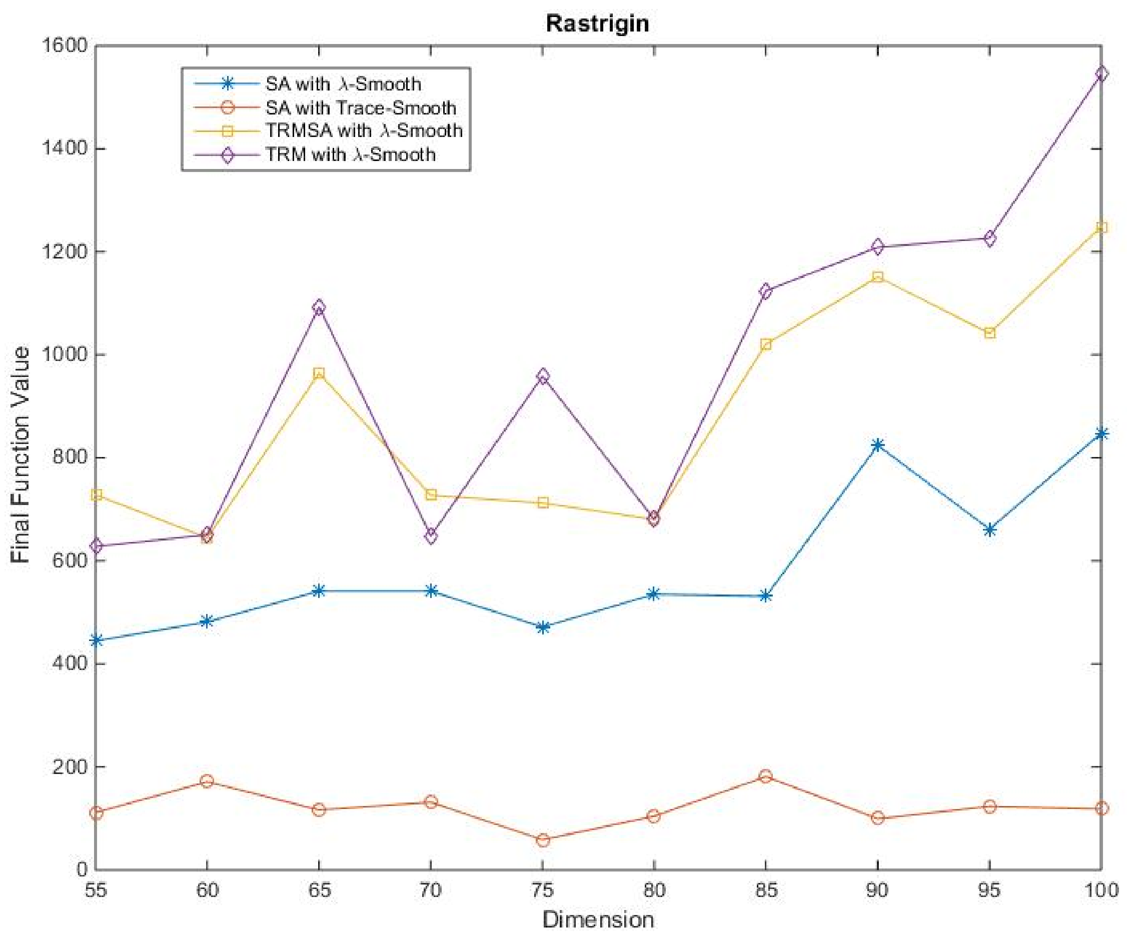


Figure 3.5: Rastrigin. 4 Methods. $n = 55$ to 100

Below we show the number of evaluation of function value and Hessian matrix. For each method, the left number shows the number of evaluation of function value and the right one is for Hessian matrix. Notice that Tr means $Trace - Smooth$ and La means $\lambda - Smooth$. We randomly generated x_* from $[-5, 5]^n$ for all $\lambda - Smooth$ method and we know the global optimum is $[0, 0, ..., 0]$.

Dim	SAwithLa		SAwithTr		TRMSAwithLa		TRMwithLa	
---	----------		-------------		-----------		-----------	
55	80412	0	80414	80412	3576	3575	627	626
60	80415	0	80412	80410	3152	3151	1116	1115
65	80412	0	80413	80411	3149	3148	1597	1596
70	80414	0	80418	80416	3528	3527	647	646
75	80416	0	80415	80413	3079	3078	1110	1109
80	80415	0	80416	80414	4533	4532	610	609
85	80413	0	80413	80411	3189	3188	679	678
90	80416	0	80415	80413	3522	3521	664	663
95	80418	0	80415	80413	3565	3564	671	670
100	80418	0	80417	80415	3287	3286	1605	1604

From the table we can see that the more information (function values and Hessian matrix) we use, the better answer we will get, which is reasonable.

Chapter 4

Summary

4.1 Advantage of SA+Smoothing

- Doing better than $fmincon$(Trust-Region) method.
- Take much less time and doing better than *simulannealbnd*(MATLAB build-in SA function).
- Take a reasonable time to get final results (With user-supply Hessian Matrix).
- The smoothing part really works.
- $\lambda - Smooth$ can work well if we know the rough region of where global optimum locates.

4.2 TRM with SA and Smoothing

- TRM with SA and Smoothing is doing better than TRM with SA and traditional TRM.
- TRM with SA and $Trace - Smooth$ does not so good as we expected since we cannot get the third and forth derivation information.
- TRM with $\lambda - Smooth$ does almost the same as TRM with SA and $\lambda - Smooth$ but needs fewer evaluations of function values and Hessian matrix.

- We can input several different x_* if we want.
- All parameters may infect the results, we should define them properly.
- For TRM with SA and Smoothing, the temperature sequence can be much shorter than SA with smoothing. For SA, usually we use $Ts = T_0 * 0.9^{(0:199)}$. For TRM with SA and Smoothing, usually we use $Ts = 10 * 0.3^{(0:4)}$
- Need some more examples and test problems. For now we use Rastrigin for most cases because for the rest of test problems we have, the results of these methods probably show no difference.

4.3 Challenges

- Δ-sequence for $Trace - Smooth$ is really important for the method, but it's a little difficult to find it perfect cause it depends on the problem. However, we have a certain way to compute it but it needs more test and proof.
- Including all challenges in simulated annealing, which means the parameter settings and so on.
- The $\lambda - Smooth$ technique can work well in the condition that we know where the global optimum locates.
- If we use $ADMAT$ to compute H, it will take a much longer time.
- SA with $Trace - Smooth$ is sensitive to the parameters.

4.4 Further Work

- Need more examples and test problems.
- How to decide the Δ or λ sequence precisely.
- For SA with $Trace - Smooth$ we may think another way to compute part of H cause we only need the trace of H.

4.5 Existing Codes

The algorithm codes for our experiments includes:

- PureSA - Pure simulated annealing algorithm.
- SmoothSA - Simulated annealing with $\lambda - Smooth$ algorithm.
- SmoothTraceSA - Simulated annealing with $Trace - Smooth$ algorithm.
- TRM - Traditional trust-region method.
- TRMSA - Trust-region method with simulated annealing.
- TRMSALaSmooth - Trust-region method with simulated annealing and $\lambda - Smooth$.
- TRMSATrSmooth - Trust-region method with simulated annealing and $Trace - Smooth$.
- TRMLaSmooth - Trust-region method with $\lambda - Smooth$.
- Trust, trust3 - Solve trust-region sub-problem, these 2 codes are from mathwork.
- neighbor - Compute neighborhood. It is from CGO package.

4.6 Test Problems and Contact Information

There are several test problems in the package, most of them come from the website: `http://www.sfu.ca/~ssurjano/optimization.html` and CGO package.

If you have any questions or concerns about this report, please contact Yichen ZHANG through yichen.zhang@uwaterloo.ca