Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 14, No. 2, pp. 327–347, 2018

ADAPTIVE INTERPOLATION STRATEGIES IN DERIVATIVE-FREE OPTIMIZATION: A CASE STUDY

W. Hare* and M. Jaberipour†

Abstract: Derivative-Free optimization (DFO) focuses on designing methods to solve optimization problems without the analytical knowledge of gradients of the objective function. There are two main families of DFO methods: model-based methods and direct search methods. In model-based DFO methods, a model of the objective function is constructed using only objective function values, and the model is used to guide the computation of the next iterate. Natural questions in this class of algorithms include how many function evaluations should be used to construct the model? And, should this number be fixed, or adaptively selected by the algorithm? In this paper, we numerically examine these questions, using Hare and Lucet's Derivative-Free Proximal Point (`DFPP`) algorithm [14] as a case study. Results suggest that the number of function evaluations used to construct the model has a huge impact on algorithm performance, and adaptive strategies can both improve and hinder algorithm performance.

Key words: *derivative-free optimization, linear interpolation, quadratic interpolation, minimum Frobenius norm, dynamic model selection*

Mathematics Subject Classification: *90C56; 49M25, 65K05*

1 Introduction

Derivative-Free optimization (DFO) focuses on designing methods to solve optimization problems without evaluating derivatives or gradients of the objective function. This is particularly applicable when the objective function is a black-box function or an oracle function, so the only available information is the value of the objective function for an input point. The study of DFO methods has grown in recent years. This is partly due to the flexibility of DFO methods across a variety of applied problems [4, 6, 15, 19, 20] (among many other examples) and partly due to the development of mathematics to ensure convergence [2, 5, 11, 14] (among many other examples). For a general overview of the DFO methods, along with a comprehensive study of many convergence results, see [10] and the many references therein.

There are two main families of DFO methods: model-based methods and direct search methods. Direct search methods, at each iteration, sample the objective function at a finite number of points and act based on those function values without any derivative approximation. A wide range variants exist, and many enjoy strong theoretical convergence analysis.

*WH research supported in part by the Natural Sciences and Engineering Research Council of Canada, Discovery Grant #355571-2013.

†MJ expresses deep gratitude to Mathematics and Computer Science, Amirkabir University of Technology, for providing a scholarship and to the Centre for Optimization, Convex Analysis and Nonsmooth Analysis at the University of British Columbia for providing supportive and stimulating working environment during his stay as a Ph.D. Visiting Scholar.

© 2018 Yokohama Publishers

However, these are not the focus of this paper.

Model-based DFO methods use past information about the objective function f to create a model function $\tilde{f}$ that approximates f. Recent years have seen a significant amount of research focused on how to construct a 'good' model function for use in DFO [8, 9, 13, 27] (among others).

Among the most common used in practice are models formed from linear interpolation [24], quadratic interpolation [7,25], or minimum Frobenius norms [12,26]. (Although, it should be noted that other techniques exist, for example radial basis function models [29,30], stochastic models [3], and models based on the Gaussian process [28].) In $\mathbb{R}^n$, linear interpolation uses $n+1$ (well-poised) points to create an approximate gradient [10, §2.3]. Alternatively, quadratic interpolation requires $(n+1)(n+2)/2$ (well-poised) points, but creates an approximate gradient and an approximate Hessian [10, §3.4]. Details on linear interpolation and quadratic interpolation are given in Section 2.

As linear interpolation only provides approximate gradients, one would expect that, in terms of the number of iterations the resulting algorithm will probably converge similar to typical first order methods (i.e., linearly). Conversely, as quadratic interpolation provides the benefit of an approximate Hessian, one might conjecture that the resulting algorithm converges similar to a second order method (i.e., quadratically). Of course, in DFO, optimizers typically measure convergence in terms of number of function calls, not in terms of iterations. (The assumption is that a function call takes significantly longer than any other portion of the algorithm.) This means, one can take $(n+2)/2$ iterations of a method using linear interpolation for every single iteration of a method using quadratic interpolation.

This led to the development and use of minimum Frobenius norm models [12, 26]. Minimum Frobenius norm models provide some balance between the extremes of linear and quadratic interpolation. Using between $n+1$ and $(n+1)(n+2)/2$ (well-poised) points, a minimum Frobenius norm model creates an approximate gradient that is more accurate than linear interpolation, along with an approximate Hessian that is less accurate than quadratic interpolation. (Details on minimum Frobenius norms are given in Section 2.) It is hoped that this balance can lead to DFO algorithms with strong convergence rates in terms of number of iterations, without the need for excessive function calls per iteration.

The availability of these three common models raises some natural questions in model-based DFO research. First, how many points should be used to create the model function, i.e., how many points should be in the *sample set*? Second, should the number of points in the sample set be static throughout the algorithm or can it be dynamically updated based on how the algorithm performed in the previous iteration? In this paper, we present the results of a case study examining these questions. Our case study focuses on the Derivative-Free Proximal Point (`DFPP`) algorithm introduced by Hare and Lucet in 2014 [14]. We test 64 basic strategies for updating the size of the sample set, and compare the strategies across 60 test problems. Numerical results suggest that the number of points in the sample set has a huge impact on algorithm performance, and adaptive strategies can both help and harm algorithm convergence.

The remainder of this paper is organized as follows. In Section 2, we provide background details on how to construct linear interpolation, quadratic interpolation, and minimum Frobenius norm models. In Section 3, we outline the `DFPP` algorithm and present our adaptive strategies for determining the size of the interpolation set. In Section 4, we discuss our numeric tests, present the results, and provide some qualitative remarks. In Section 5, we provide some conclusions. Tables of results appear in the appendix.

Throughout, $B_\Delta(y^0)$ denotes the closed ball of radius Δ centred at y^0: $B_\Delta(y^0) = \{x : \|x - y^0\| \leq \Delta\}$,

2 Model Construction Techniques

Let $\mathcal{P}_n^d$ be the space of polynomials of degree less than or equal to d in $\mathbb{R}^n$. For $d = 1$ and $d = 2$, the dimension of this space is $\dim(\mathcal{P}_n^1) = n + 1$ and $\dim(\mathcal{P}_n^2) = \frac{(n+1)(n+2)}{2}$, respectively. A basis $\Phi = \{\phi_1, ..., \phi_q\}$ of $\mathcal{P}_n^d$ is a set of q polynomials of degree less than or equal to d such that $q = \dim(\mathcal{P}_n^d)$ and the polynomials span $\mathcal{P}_n^d$. If Φ is a basis in $\mathcal{P}_n^d$, then any polynomial $m \in \mathcal{P}_n^d$ can be formulated as $m(x) = \sum_{j=1}^q \alpha_j \phi_j(x)$, where $\{\alpha_1, \alpha_2, ...\alpha_q\}$ is a uniquely determined set of real coefficients.

We say the polynomial m interpolates the function f at a given point y if $m(y) = f(y)$. Suppose we are given a set $Y = \{y^0, y^1, ..., y^p\}$ of interpolation points, and we seek a polynomial, m, with degree less than or equal to d that interpolates a given function f at the points in Y. Since it must be possible to write m in the form of $\sum_{j=1}^q \alpha_j \phi_j$, we seek interpolation coefficients, α_j, that satisfy the interpolation conditions

$$m(y^i) = \sum_{j=1}^{q} \alpha_j \phi_j(y^i) = f(y^i), i = 0, ..., p. \tag{2.1}$$

Conditions (2.1) form a linear system in terms of the interpolation coefficients, which we will write in matrix form as $M(\Phi, Y)\alpha_\phi = f(Y)$, where

$$M(\Phi, Y) = \begin{pmatrix} \phi_0(y^0) & \phi_1(y^0) & \dots & \phi_q(y^0) \\ \phi_0(y^1) & \phi_1(y^1) & \dots & \phi_q(y^1) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_0(y^p) & \phi_1(y^p) & \dots & \phi_q(y^p) \end{pmatrix},$$

$$\alpha_\phi = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_q \end{pmatrix}, \quad \text{and} \quad f(Y) = \begin{pmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^p) \end{pmatrix}.$$

If conditions (2.1) have a unique solution, then their solution provides m. If conditions (2.1) have multiple solutions, then m is said to be under determined, and we must impose additional conditions to select m. If conditions (2.1) have no solution, then m cannot exist, but a least-squares solution could be used to find an approximate solution and create an m that approximates f in the sense of statistical regression.

2.1 Linear Interpolation

Linear interpolation sets the maximum degree of the polynomial to $d = 1$; i.e., linear interpolation applies in $\mathcal{P}_n^1$. The natural basis for this space is $\Phi = \{1, x_1, x_2, ...x_n\}$. Our interpolation conditions (equation (2.1)), can be simplified to

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

where

$$M(Y) = \begin{pmatrix} 1 & y_1^0 & \dots & y_n^0 \\ 1 & y_1^1 & \dots & y_n^1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & y_1^n & \dots & y_n^n \end{pmatrix}, \quad \alpha = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{pmatrix}, \quad \text{and} \quad f(Y) = \begin{pmatrix} f(y^0) \\ f(y^1) \\ \vdots \\ f(y^n) \end{pmatrix}.$$

Clearly, $M(Y)$ is invertible if and only if conditions (2.1) have a unique solution. More importantly, the error in the gradient approximation from linear interpolation can be quantified in terms of several constants and the approximate radius of the sample set.

Theorem 2.1 ((Error bound for Linear interpolation) [10, Thm 2.11])**.** *Let $Y = \{y^0, y^1, \dots, y^n\} \subseteq \mathcal{R}^n$ be poised for linear interpolation. Define $\Delta = \Delta(Y) = \max_{1\leq i\leq n} \|y^i - y^0\|$. Suppose the function f is continuously differentiable in an open domain Ω containing $B_\Delta(y^0)$, and ∇f is Lipschitz continuous in Ω with constant $\nu > 0$. Let m be the linear function that interpolates f over all points in Y. Then for all points $y \in B_\Delta(y^0)$, we have*

$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\Delta,$$

where κ_{eg} is a constant based on ν, n, and the geometry of the interpolation set.

2.2 Quadratic Interpolation

Quadratic interpolation sets the maximum degree of the polynomial to $d = 2$; i.e., quadratic interpolation applies in $\mathcal{P}_n^2$. One natural basis for quadratic interpolation is

$$\Phi = \{1, x_1, x_2, ...x_n, \frac{1}{2}(x_1)^2, x_1x_2, ...x_1x_n, \frac{1}{2}(x_2)^2, x_2x_3, ...\frac{1}{2}(x_n)^2\}.$$

Using this basis, one can again write conditions (2.1) as a linear system. (For the sake of space, we do not rewrite the system here.) The system results in the matrix $M(\Phi, Y)$ being a $(n+1)(n+2)/2 \times (n+1)(n+2)/2$ square matrix. Like linear interpolation, the error in the gradient approximation from quadratic interpolation can be quantified using several constants and the approximate radius of the sample set.

Theorem 2.2 ((Error bounds for quadratic interpolation) [10, Thm 3.16])**.** *Let $Y = \{y^0, y^1, \dots, y^p\} \subseteq \mathcal{R}^n$ be poised for quadratic interpolation. Define $\Delta = \Delta(Y) = \max_{1\leq i\leq p} \|y^i - y^0\|$. Suppose the function f is twice continuously differentiable in an open domain Ω containing $B_\Delta(y^0)$, and $\nabla^2 f$ is Lipschitz continuous in Ω with constant $\nu_2 > 0$. Let m be the quadratic function that interpolates f over all points in Y. Then, for all points $y \in B_\Delta(y^0)$, we have*

$$\|\nabla^2 f(y) - \nabla^2 m(y)\| \leq \kappa_{eh}\Delta, \text{ and}$$
$$\|\nabla f(y) - \nabla m(y)\| \leq \kappa_{eg}\Delta^2,$$

where κ_{eh} and κ_{eg} are constants based on ν_2, p, and the geometry of the interpolation set.

2.3 Minimum Frobenius Norm Models

Minimum Frobenius norm models set the maximum degree of the polynomial to $d = 2$, but work in the case when the number of interpolation points is less than the $(n+1)(n+2)/2$ required for quadratic interpolation.

In this case, the $M(\Phi, Y)$ defining the interpolating conditions has more columns than rows and the interpolation polynomials are no longer unique.

Let us split the natural basis Φ into linear and quadratic parts: $\Phi_L = \{1, x_1, ..., x_n\}$ and $\Phi_Q = \{\frac{1}{2}x_1^2, x_1x_2, ..., \frac{1}{2}x_n^2\}$. The interpolation model can now be written as

$$m(x) = \alpha_L^T \Phi_L(x) + \alpha_Q^T \Phi_Q(x),$$

where α_L and α_Q are the appropriate parts of the coefficient vector α.

In a DFO framework with under determined interpolation, it is desirable to construct accurate linear models and then enhance them with curvature information, hoping that the actual accuracy of the model is better than that of a purely linear model. (Hence, it is important to construct sample sets that are poised for linear interpolation.)

Since the interpolation set is too small to create a unique quadratic interpolation, we must impose some additional requirements to determine the final model. As our Hessian approximation will be of a lower accuracy than our gradient approximation, in derivative-free optimization it makes sense to seek a model for which the norm of the Hessian is small or moderate. Therefore, we define the minimum Frobenius norm solution as a solution to the following optimization problem in α_L and α_Q:

$$\min \frac{1}{2} \parallel \alpha_Q \parallel_2^2$$

$$M(\Phi_L, Y)\alpha_L + M(\Phi_Q, Y)\alpha_Q = f(Y). \tag{2.2}$$

The name minimum Frobenius norm solution comes from the equivalence of minimizing the norm of α_Q and minimizing the Frobenius norm of the Hessian of m.

The condition for the existence and uniqueness of the minimum Frobenius norm model is that the following matrix is nonsingular

$$F(\Phi, Y) = \begin{pmatrix} M(\Phi_Q, Y)M(\Phi_Q, Y)^T & M(\Phi_L, Y) \\ M(\Phi_L, Y)^T & 0 \end{pmatrix}. \tag{2.3}$$

We say that a set Y is poised for minimum Frobenius norm interpolation if problem (2) has a unique solution or, equivalently, if the matrix $F(\Phi, Y)$ is non-singular. Like linear and quadratic interpolation, the error bounds of the resulting model can be bounded using constants and the approximate radius of the sample set.

Theorem 2.3 ((Error bounds for minimum Frobenius norm models) [10, Thm 5.4])**.** *Let $Y = \{y^0, y^1, \ldots, y^p\} \subseteq \mathcal{R}^n$ be poised for minimum Frobenius norm interpolation. Define $\Delta = \Delta(Y) = \max_{1 \le i \le p} \parallel y^i - y^0 \parallel$. Suppose the function f is continuously differentiable in an open domain Ω containing $B_\Delta(y^0)$, and ∇f is Lipschitz continuous in Ω with constant $\nu > 0$. Let m be the quadratic function that results from the minimum Frobenius norm interpolation of f over all points in Y. Then, for all points $y \in B_\Delta(y^0)$, we have*

$$\|\nabla f(y) - \nabla m(y)\| \le \kappa_{eg}\Delta,$$

and κ_{eg} is a constants based on ν, p, the geometry of the interpolation set, and the norm of the model Hessian.

3 Derivative-Free Proximal Point Method

Much like Newton's method is a standard tool for solving smooth optimization problems, proximal point algorithms can be viewed as an analogous tool for nonsmooth optimization. The basic (theoretical) method solves the minimization of f through iterative solutions to the proximal point problem

$$x^{k+1} = \text{prox}_r f(x^k) \quad \text{where } \text{prox}_r f(x^k) = \arg\min\{f(y) + \frac{r}{2}\|y - x^k\|^2\}. \tag{3.1}$$

In practice, it is unnecessary to solve $\text{prox}_r f(x^k)$ exactly, which has lead to a variety of practical implementations based on the proximal point framework [17,18,23] (and references therein).

Most common are the proximal-bundle methods, where the objective function f is replaced by a sequence of piecewise linear model functions f_k, see [17,18,23]. Such methods essentially replace the minimization of f with a sequence of quadratic programming problems.

Recently, Hare and Lucet introduced a Derivative-Free Proximal Point (`DFPP`) method [14]. Within the framework, x^k denotes the prox-centre of the algorithm during iteration k. At each iteration, the algorithm shall make use of a sample set $Y = \{y^0, y^1, ..., y^p\} \subseteq \mathbb{R}^n$ with $y^0 = x^k$ to construct a model of the objective function. In the algorithm, the approximate sampling radius of Y is defined $\Delta(Y) = \max_{y^i \in Y} \|y^i - y^0\|$, and $\lambda_n(H)$ is used to denote the minimum eigenvalue of H. Pseudo-code of the `DFPP` algorithm follows.

Derivative-Free Proximal Point Method (DFPP)

0. INITIALIZE: Set $k = 0$ and input

    x^0 - an initial prox-centre,
    Y^0 - an initial poised interpolation set with $x^0 \in Y^0$,
    r^0 - an initial prox-parameter, $r^0 > 0$
    m - an Armijo-like parameter, $0 < m < 1$,
    Γ - a minimal radius decrease parameter, $0 < \Gamma < 1$,
    $r_{\texttt{tol}}$ - stopping tolerance for prox-parameter, $r_{\texttt{tol}} > 0$,
    $\Delta_{\texttt{tol}}$ - stopping tolerance for search radius, $\Delta_{\texttt{tol}} \geq 0$, and
    $\varepsilon^{\nabla}_{\texttt{tol}}$, $\varepsilon^{\Delta}_{\texttt{tol}}$ - stopping tolerances for approximate gradient, $\varepsilon^{\nabla}_{\texttt{tol}}, \varepsilon^{\Delta}_{\texttt{tol}} > 0$.

1. MODEL AND STOPPING CONDITIONS:
    Create q^k, a model of f over Y^k:
    $$q^k(x) := a^k + \langle g^k, x\rangle + \frac{1}{2}\langle x, H^k x\rangle.$$
    If $\|\nabla q^k(x^k)\| < \varepsilon^{\nabla}_{\texttt{tol}}$ and $\Delta(Y^k) < \varepsilon^{\Delta}_{\texttt{tol}}$, then STOP ('success').
    If $\Delta(Y^k) < \Delta_{\texttt{tol}}$, then STOP.

2. PROX-FEASIBLITY CHECK:
    If $r^k \leq -\lambda_n(H^k)$, then ($q^k + r^k \frac{1}{2}\|\cdot\|^2$ is not strictly convex):
    - reset $r^k = -\lambda_n(H^k) + 1$,

3. PROX TRIAL POINT:
    Compute the trial point
    $$\{\tilde{x}^k\} = \text{prox}_{r^k} q^k(x^k) = \{(H^k + r^k \text{Id})^{-1}(r^k x^k - g^k)\}.$$
    Compute the predicted decrease
    $$\delta^k = q^k(x^k) - q^k(\tilde{x}^k).$$

4. SERIOUS/NULL CHECK:
    If $f(\tilde{x}^k) \leq f(x^k) - m\delta^k$, then declare a *serious step*:
    - select x^{k+1} such that $f(x^{k+1}) \leq f(x^k) - m\delta^k$,

– generate an interpolation set Y^{k+1} such that $x^{k+1} \in Y^{k+1}$ and $\Delta(Y^{k+1}) \leq \Delta(Y^k)$.

Else (if $f(\tilde{x}^k) > f(x^k) - m\delta^k$), then declare a *null step*:

– if $\tilde{x}^k \notin B_{\Delta(Y^k)}(x^k)$, then declare the null step to be *type 1*, increase $r^{k+1} \to 2r^k$, set $x^{k+1} = x^k$ and Y^{k+1} with $Y^k \subseteq Y^{k+1}$ and $\Delta(Y^{k+1}) = \Delta(Y^k)$,

– if $\tilde{x}^k \in B_{\Delta(Y^k)}(x^k)$, then declare the null step to be *type 2*, set $x^{k+1} = x^k$ and generate an interpolation set Y^{k+1} such that $x^{k+1} \in Y^{k+1}$ and $\Delta(Y^{k+1}) \leq \Gamma\Delta(Y^k)$.

5. Loop:

   Increment $k \to k+1$ and return to Step 1.

3.1 Adaptive Strategies in DFPP

The DFPP framework has two interesting features that make it well-suited to exploring adaptive updating of the number of points in the sample set at each iteration.

First, in order for the algorithm to converge the model functions q^k must satisfy the following assumption (see [14]).

Assumption 3.1. Assume $f \in \mathcal{C}^1$. Furthermore, assume that there exists constants C and M such that, for any point y^0 and any sampling radius $\Delta > 0$, we are able to generate a sampling set $Y = \{y^0, y^1, \dots y^p\} \subseteq \mathbb{R}^n$ and a corresponding quadratic model function q such that $\Delta(Y) = \Delta$ and

$$\begin{array}{rcll} \|f(y) - q(y)\| & \leq & C\Delta^2 & \text{for all } y \in B_\Delta(y^0), \\ \|\nabla f(y) - \nabla q(y)\| & \leq & C\Delta & \text{for all } y \in B_\Delta(y^0), \text{ and} \\ \|\nabla^2 q(y)\| & \leq & M. & \end{array}$$

Note that the Assumption 3.1 can be satisfied through

a) a linear interpolation model, by noting $\nabla^2 q = 0$ in this case,

b) a quadratic interpolation model, see [14, Lem 3.1], or

c) a minimum Frobenius norm model, see [14, p. 209].

Moreover, the algorithm can use a different model construction technique at each iteration, without compromising the convergence analysis.

The second interesting feature of the DFPP framework is that the algorithm ends step 4 with one of three possible declarations: *serious step, null step type 1,* or *null step type 2.* In a *serious step*, the algorithm was successful in finding a new proximal centre, which shows notable decrease over the previous proximal centre. In a *null step type 1*, the algorithm was unable to find a new proximal centre and the predicted new centre was outside of the radius of accuracy for the model. In this case, the prox-parameter is increased, but the old sample set can be reused (possibly with additional points added). Finally, in a *null step type 2*, the algorithm was unable to find a new proximal centre, despite the fact that the predicted new centre was inside of the radius of accuracy for the model. In this case, the desired accuracy of the model is increased (i.e., $\Delta(Y)$ is decreased), and a new model must be constructed.

As each outcome suggests a different situation, and each iteration can use a different model construction technique, the `DFPP` algorithm naturally lends itself to the idea of using an adaptive strategy for selecting sample set size at each iteration. The adaptive strategies explored in this paper can be viewed as selecting the number of points in the sample set in the next iteration based on the declaration in step 4:

i. if step 4 of the `DFPP` method declares a serious step, then use a sample set of size $N_{\mathtt{s}}$ in the next iteration,

ii. if step 4 of the `DFPP` method declares a null step type 1, then use a sample set of size $N_{\mathtt{n1}}$ in the next iteration, and

iii. if step 4 of the `DFPP` method declares a null step type 2, then use a sample set of size $N_{\mathtt{n2}}$ in the next iteration.

For our testing, $N_{\mathtt{s}}, N_{\mathtt{n1}}$, and $N_{\mathtt{n2}}$ are each taken from

$$\left\{n+1, 2n+1, \left\lfloor \frac{1}{2}\left((n+1) + \frac{(n+1)(n+2)}{2}\right)\right\rfloor, \frac{(n+1)(n+2)}{2}\right\},$$

where n is the problem dimension. In order to simplify presentation, we use the following notation for these four strategies

$$\begin{aligned} \mathtt{S_n} &= n+1, \\ \mathtt{S_{2n}} &= 2n+1, \\ \mathtt{S_{n^2/4}} &= \left\lfloor \tfrac{1}{2}\left((n+1) + \tfrac{(n+1)(n+2)}{2}\right)\right\rfloor, \\ \mathtt{S_{n^2/2}} &= \tfrac{(n+1)(n+2)}{2}. \end{aligned}$$

As there are 3 possible ways to conclude step 4, and we examine 4 different strategies for each conclusion, we explore a total of $4^3 = 64$ different adaptive strategies.

3.2 Sample Set Construction Techniques

The error bounds provided in Theorems 2.1, 2.2, and 2.3, all rely on the "geometry of the interpolation set". This phrase actually hides a deep literature on the topic. While, in order for the sample set to be poised, we require an appropriate matrix, $M(\Phi, Y)$ or $F(\Phi, Y)$, to be invertible, in practice it is important that this matrix is 'stable'. This stability is dependent on the "geometry of the interpolation set". Details on quantifying the geometry of the interpolation set, and how to control the quality of this geometry, are outside of the scope of this work (we refer interested readers to [10, Chpt 2–6]). Nonetheless, some comments are in order.

For the numerical testing in this paper, we use Algorithms 6.2 and 6.3 of [10] to construct our interpolation sets and improve their geometry. The interpolation set Y^k is built based on three possible declarations in step 4.

i. If step 4 of the `DFPP` method declares a serious step, then
   - place x^{k+1} into an interpolation set Y in position of $x = y_0$,
   - place any previously sampled points in $B_{\Delta(Y^k)}(x^{k+1})$ into Y,
   - use [10, Alg 6.2 & 6.3] to expand Y into a well-poised set of $(n+1)(n+2)/2$ points, and
   - select $N_{\mathtt{s}}$ points from Y to define Y^{k+1}.

ii. If step 4 of the DFPP method declares a null step type 1, then

- if the number of points in Y^k is greater or equal to $N_{\mathtt{n1}}$, then select $N_{\mathtt{n1}}$ points from Y^k to define Y^{k+1},
- if the number of points in Y^k is less than $N_{\mathtt{n1}}$, then use [10, Alg 6.2 & 6.3] to create a well-poised set of $(n+1)(n+2)/2$ points and select $N_{\mathtt{n1}}$ points from it to define Y^{k+1}.

iii. If step 4 of the DFPP method declares a null step type 2, then

- place x^{k+1} into an interpolation set Y in position of $x = y_0$,
- place any previously sampled points in $B_{\Gamma\Delta(Y^k)}(x^{k+1})$ into Y,
- use [10, Alg 6.2 & 6.3] to expand Y into a well-poised set of $(n+1)(n+2)/2$ points, and
- select $N_{\mathtt{n2}}$ points from Y to define Y^{k+1}.

In all of the cases above, the selection of the final subset of $N_{\mathtt{x}}$ points from Y is done by taking the first $N_{\mathtt{x}}$ points, and then a safety check is used to ensure the final interpolation set is well-poised. If it is not, then we use [10, Alg 6.2 & 6.3] to create a well-poised set of $(n+1)(n+2)/2$ points and select the first $N_{\mathtt{x}}$ points from it to define Y^{k+1} (again with a safety check to ensure well-poised).

4 Numerical Results

The DFPP method is implemented in MATLAB [14]. Minor adaptations to the original code allowed for the adaptive strategies in Subsection 3.1 to be incorporated. Minor tuning to select algorithmic parameters was performed. Specifically, Armijo-like parameters $m \in \{0.1, 0.5, 0.9\}$ were tested and radius decrease parameters $\Gamma \in \{0.25, 0.5\}$ were tested. The values $m = 0.1$ and $\Gamma = 0.5$ provided the best overall performance across all strategies. The initial prox-parameter was set to $r^0 = 1$. (As improvement based on these parameters was extremely minor, we do not present results from other parameter combinations; however, these results are available through contacting the corresponding author.) Finally, in Step 4, if a serious step is declared, the user has the option of performing a line search (or other search method) to seek x^{k+1} that provides some further improvement over $\tilde{x}^k$. We tested using no additional searching and using a backtracking line search.

The strategies were tested on the 60 problems from [1, 16, 21]. Test problems were separated into two groups: low dimension and high dimension. Table 1 lists the name and the dimension of each test problem.

For each test problem, each strategy was run until a total of $100n$ function calls was exceeded, where n is the dimension of the test problem.

In order to rank the strategies, we consider the following *improvement* metric

$$\mathtt{imp}(N_{\mathtt{s}}, N_{\mathtt{n1}}, N_{\mathtt{n2}}) = \sum_{\mathcal{P}} \min\left\{ -\log_{10} \frac{|(f - fbest)|}{|(f_0 - fbest)|}, 16 \right\}$$

where $\mathcal{P}$ is the set of all test problems, f is the objective function value obtained by DFPP, $fbest$ is the best known objective function value, and f_0 is the initial objective function value. The value $-\log_{10} \frac{|(f-fbest)|}{|(f_0-fbest)|}$ can loosely be interpreted as the number of new digits of accuracy (in function value) obtained on a given test problem. The minimization with 16 deals with the (few) problems that end up being solved exactly and return unrealistic values like $-\log_{10}\frac{|(f-fbest)|}{|(f_0-fbest)|} > 1000$. Without capping, these problems can massively skew the data analysis. Finally, these values are summed over all test problems, to give each strategy a total improvement.

The aggregate results when no line search was used appear in Tables 2 to 4 (in the Appendix) and the aggregate results when a backtracking line search was used appear in Tables 5 to 7 (in the Appendix). Tables 2 and 5 provide the results when all test problems are considered. Tables 3 and 6 provide the results when only low dimension test problems are considered. Tables 4 and 7 provide the results when only high dimension test problems are considered.

4.1 Interpretation of the Results

To ease interpretation, Tables 2 to 7 are sorted from the highest `imp` value to the lowest `imp` value.

Examining Tables 2 and 5 (which contain all test problems grouped), we note that the line search has a strong positive impact on the performance of the algorithm. This was also noted in [14]. In fact, the line search is so effective that the worst result in Table 5, would rank 7^{th} if it were placed in Table 2.

Across all tables we see a common trend of $N_{\mathtt{s}} = \mathtt{S_n}$. That is, if a serious step occurred, then the next model should be as simple to create as possible. This makes sense, as serious steps correspond with success and movement of the prox-centre. If a serious step occurred, then the next model essentially starts from scratch, so it makes sense to build a simple model and only increase complexity if the next iteration induces a null step.

Comparing low dimension to high dimension problems presents some enlightening results. In both Tables 3 and 6 (low dimensions), we see 4 of the top 5 strategies involve building complex models when a null step occurs (i.e., $N_{\mathtt{n1}} \in \{\mathtt{S_{n^2/4}}, \mathtt{S_{n^2/2}}\}$ or $N_{\mathtt{n2}} \in \{\mathtt{S_{n^2/4}}, \mathtt{S_{n^2/2}}\}$). Conversely, in Tables 4 and 7 (high dimensions), we see complex models are generally avoided: in Table 4, $N_{\mathtt{n1}} \in \{\mathtt{S_n}, \mathtt{S_{2n}}\}$ and $N_{\mathtt{n2}} \in \{\mathtt{S_n}, \mathtt{S_{2n}}\}$ for all of the top 5, while in Table 7, $N_{\mathtt{n1}} \in \{\mathtt{S_n}, \mathtt{S_{2n}}\}$ and $N_{\mathtt{n2}} \in \{\mathtt{S_n}, \mathtt{S_{2n}}\}$ for 3 of the top 5.

4.2 Data Profiles

While Tables 2 and 5 provide some insight into the performance, they rely strongly on data aggregation. As such, it is possible that certain problems are being solved to very high precision and skewing the results. In this subsection we present data profiles [22] of select strategies.

Data profiles are designed to capture both speed and robustness of a solver, by plotting the portion of problems solved using less than or equal to $\alpha \times n + 1$ function calls, where α is the number of 'simplex gradients equivalents' used and $n + 1$ represent the number of function calls required to create a simplex gradient in $\mathbb{R}^n$. For further details on data profiles, we refer the reader to [22].

With 128 strategies tested (64 adaptive approaches times 2 for the use/disuse of a line search), presenting all data profiles would result in an unreadable figure. Instead, we present data profiles containing:

- the 'most basic' non-adaptive strategy – $N_{\mathtt{s}} = N_{\mathtt{n1}} = N_{\mathtt{n2}} = \mathtt{S_n}$,
- the 'most complex' non-adaptive strategy – $N_{\mathtt{s}} = N_{\mathtt{n1}} = N_{\mathtt{n2}} = \mathtt{S_{n^2/2}}$, and
- the top two adaptive strategies – $N_{\mathtt{s}} = \mathtt{S_n}$, $N_{\mathtt{n1}} = \mathtt{S_n}$, $N_{\mathtt{n2}} = \mathtt{S_{2n}}$, and $N_{\mathtt{s}} = \mathtt{S_n}$, $N_{\mathtt{n1}} = \mathtt{S_{2n}}$, $N_{\mathtt{n2}} = \mathtt{S_n}$.

For more detailed results please contact the corresponding author.

Data profiles are created including the no line search and with a line search option. The data profile for a solving tolerance of 10^{-3} appears in Figure 1 and data profile for a solving tolerance of 10^{-6} appears in Figure 2.

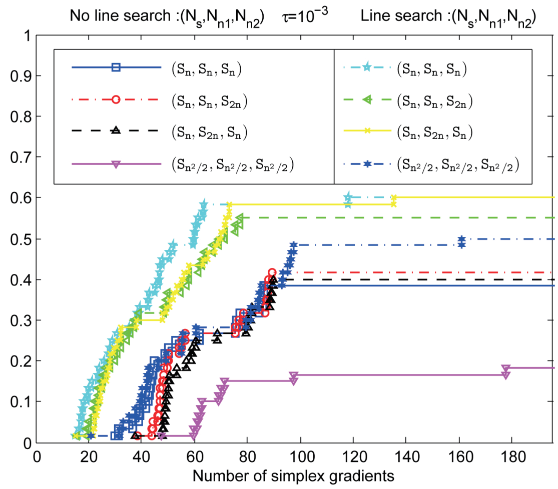

Figure 1: Data profile for 8 strategies and a solving tolerance of 10^{-3}.

Figure 1 and 2 show the expected results and a few surprising results. First, it is again clear that the line search provides an excellent performance boost to the algorithm. Examining just methods that use a line search, we note that strategy $N_{\mathtt{s}} = N_{\mathtt{n1}} = N_{\mathtt{n2}} = \mathtt{S_n}$ outperforms all other methods. However, examining the no line search methods, we see that for solving tolerance of 10^{-3} strategy $N_{\mathtt{s}} = N_{\mathtt{n1}} = \mathtt{S_n}$, $N_{\mathtt{n2}} = \mathtt{S_{2n}}$ outperforms the other no line search methods. Meanwhile, for solving tolerance of 10^{-6} strategy $N_{\mathtt{s}} = N_{\mathtt{n2}} = \mathtt{S_n}$, $N_{\mathtt{n1}} = \mathtt{S_{2n}}$ outperforms the other no line search methods. Neither of these victories are resounding, but it nonetheless suggests that adaptive strategies may have some place in future DFO algorithms.

5 Conclusions

Model-based DFO methods work by constructing local models of the objective function using a set of function evaluations. In this paper, we explore the questions of how many function evaluations should be used to construct the model, and should this number be fixed, or adaptively selected by the algorithm? We approach the question numerically, by making use of the flexibility and iteration decision structure within the `DFPP` algorithm of [14]. The results suggest that, for this algorithm, and this implementation, adaptive strategies can provide some improvement, particularly in higher dimensions. However, the results also show that a poorly selected adaptive strategy can greatly hinder performance, both in low and high dimensions. Finally, the results generally suggest that, for this algorithm and implementation, basic models using fewer function evaluations outperform complex models that require many function evaluations to build.

It should be noted that there are many model-building methods that were not considered in this paper: e.g., linear regression models, centered simplex gradients, radial basis functions, models based on the Gaussian process, etc. Also, while past points within the

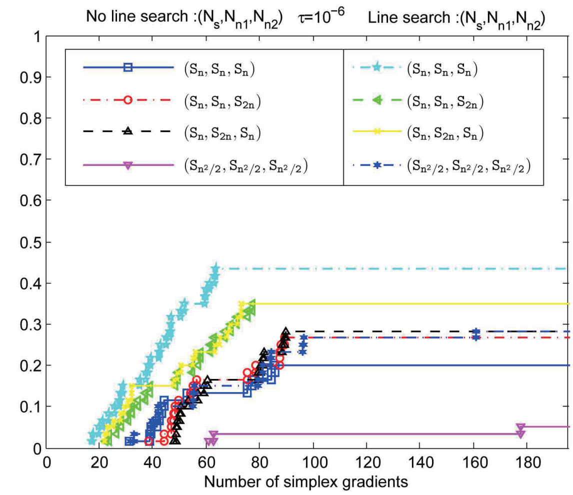

Figure 2: Data profile for 8 strategies and a solving tolerance of 10^{-6}.

sampling radius were used when building new models, advance techniques on minimum Frobenius norm based model updating was not applied within this paper. This leaves significant opportunity for further research in this area.

References

[1] M.M. Ali, C. Khompatraporn and Z.B. Zabinsky, A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, *Journal of Global Optimization* 31 (2005) 635–672.

[2] C. Audet and J.E. Dennis, Jr, Analysis of generalized pattern searches, *SIAM J. Optim.* 13 (2003) 889–903 (electronic).

[3] A.S. Bandeira, K. Scheinberg and L.N. Vicente, Convergence of trust-region methods based on probabilistic models. *SIAM J. Optim.* 24 (2014) 1238–1264.

[4] K. Bigdeli, W. Hare and S. Tesfamariam, Optimal design of viscous damper connectors for adjacent structures using genetic algorithm and Nelder-Mead algorithm, in *Proceeding of SPIE conference on Smart Structures and Materials*, 2012.

[5] C. Bogani, M.G. Gasparo and A. Papini, Generating set search methods for piecewise smooth problems, *SIAM J. Optim.* 20 (2009) 321–335.

[6] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini and V. Torczon, Optimization using surrogate objectives on a helicopter test example, in *Computational methods for optimal design and control (Arlington, VA, 1997)*, Progr. Systems Control Theory, vol. 24, Birkhäuser Boston, Boston, MA, 1998, pp. 49–58.

[7] A. Conn, K. Scheinberg and P. Toint, On the convergence of derivative-free methods for unconstrained optimization, in *Approximation theory and optimization (Cambridge, 1996)*, pages 83–108. Cambridge Univ. Press, Cambridge, 1997, pp. 83–108.

[8] A.R. Conn, K. Scheinberg and L.N. Vicente, Geometry of interpolation sets in derivative free optimization, *Math. Program. Ser. B* 111 (2008) 141–172.

[9] A.R. Conn, K. Scheinberg and L.N. Vicente, Geometry of sample sets in derivative-free optimization: polynomial regression and underdetermined interpolation, *IMA J. Numer. Anal.* 28 (2008) 721–748.

[10] A.R. Conn, K. Scheinberg and L.N. Vicente. *Introduction to derivative-free optimization*, MPS/SIAM Series on Optimization, vol. 8, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009.

[11] A.L. Custódio, J.E. Dennis, Jr. and L.N. Vicente, Using simplex gradients of nonsmooth functions in direct search methods, *IMA J. Numer. Anal.* 28 (2008) 770–784.

[12] A.L. Custódio, H. Rocha and L.N. Vicente, Incorporating minimum Frobenius norm models in direct search, *Comput. Optim. Appl.* 46 (2010) 265–278.

[13] E.A.E. Gumma, M.H.A. Hashim and M.M. Ali, A derivative-free algorithm for linearly constrained optimization problems, *Comput. Optim. Appl.* 57 (2014) 599–621.

[14] W. Hare and Y. Lucet, Derivative-free optimization via proximal point methods, *J. Optim. Theory App.* 160 (2014) 204–220.

[15] W.L. Hare, Using derivative free optimization for constrained parameter selection in a home and community care forecasting model, in *International Perspectives on Operations Research and Health Care, Proceedings of the 34th Meeting of the EURO Working Group on Operational Research Applied to Health Sciences*, 2010, pp. 61–73.

[16] A. Hedar, Test functions for unconstrainted global optimization, `http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm`.

[17] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms. I*, *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*, vol. 305, Springer-Verlag, Berlin, 1993.

[18] M.M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optim. Methods Softw.* 17 (2002) 1–29.

[19] A.L. Marsden, J.A. Feinstein and C.A. Taylor, A computational framework for derivative-free optimization of cardiovascular geometries, *Comput. Methods Appl. Mech. Engrg.* 197 (2008) 1890–1905.

[20] A.L. Marsden, M. Wang, J.E. Dennis, Jr. and P. Moin, Trailing-edge noise reduction using derivative-free optimization and large-eddy simulation, *J. Fluid Mech.* 572 (2007) 13–36.

[21] J.J. Moré, B.S. Garbow and K.E. Hillstrom, Testing unconstrained optimization software, *ACM Trans. Math. Software* 7 (1981) 17–41.

[22] J.J. Moré and S.M. Wild, Benchmarking derivative-free optimization algorithms, *SIAM J. Optim.* 20 (1999) 172–191.

[23] W. Oliveira and C. Sagastizábal, Bundle methods in the XXIst century: A bird's-eye view, *Pesquisa Operacional* 34 (2014) 647– 670.

[24] M.J.D. Powell, A direct search optimization method that models the objective and constraint functions by linear interpolation, in *Advances in optimization and numerical analysis (Oaxaca, 1992)*, *Math. Appl.*, vol. 275, Kluwer Acad. Publ., Dordrecht, 1994, pp. 51–67.

[25] M.J.D. Powell, Uobyqa: unconstrained optimization by quadratic approximation, *Math. Program.* 92 (2002) 555–582.

[26] M.J.D. Powell, Least Frobenius norm updating of quadratic models that satisfy interpolation conditions, *Math. Program. Ser. B* 100 (2004) 183–215.

[27] K. Scheinberg and Ph.L. Toint, Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization, *SIAM J. Optim.* 20 (2010) 3512–3532.

[28] W. Scott, P. Frazier and W. Powell, The correlated knowledge gradient for simulation optimization of continuous parameters using Gaussian process regression, *SIAM J. Optim.* 21 (2011) 996–1026.

[29] S.M. Wild, R.G. Regis, and C.A. Shoemaker, ORBIT: optimization by radial basis function interpolation in trust-regions, *SIAM J. Sci. Comput.* 30 (2008) 3197–3219.

[30] W. Yao, X.Q. Chen, Y.Y. Huang and M. van Tooren, A surrogate-based optimization method with RBF neural network enhanced by linear interpolation and hybrid infill strategy, *Optim. Methods Softw.* 29 (2014) 406–429.

Manuscript received 2 April 2015
revised 7 August 2015, 20 October 2015
accepted for publication 7 November 2015

W. Hare
Mathematics, Irving K. Barber School
University of British Columbia
Okanagan Campus, Kelowna
B.C., V1V 1V7, Canada
E-mail address: warren.hare@ubc.ca

M. Jaberipour
Mathematics and Computer Science
Amirkabir University of Technology
Tehran, Iran
E-mail address: majid.jaberipour@gmail.com

A Tables

Table 1: Test problems.

Low Dim. Problems	
Function Name	Dim.
Bard	3
Beale	2
Biggs EXP6	6
Box 3D	3
Brown almost-linear	3
Brown & Dennis	4
Brown badly scaled	2
Broyden banded	3
Broyden tridiagonal	3
Discrete boundary value	3
Discrete integral eq.	3
Freudenstein & Roth	2
Gaussian	3
Gulf	3
Helical valley	3
Jennrich & Sampson	2
Kowalik & Osborne	4
Linear rank-1	3
Linear rank full	4
Linear rank-1 with zeros	3
Meyer	3
Osborne I	5
Penalty I	4
Penalty II	4
Powell badly scaled	2
Rosenbrock	2
Trigonometric	3
Variably dimensional	3
Watson	6
Wood	4

High Dim. Problems	
Function Name	Dim.
Ackley	10
Ackley	20
Arrowhead	10
Arrowhead	20
Epistatic Michalewicz	10
Exponential	10
Exponential	20
Griewank	10
Griewank	20
Levy & Montalvo I	10
Levy & Montalvo I	20
Levy & Montalvo II	10
Levy & Montalvo II	20
Modified Langerman	10
Neumaier 3	10
Neumaier 3	20
Powell singular	12
Powell singular	20
Rastrigin	10
Rastrigin	20
Rosenbrock	10
Rosenbrock	20
Sinusoidal	10
Sinusoidal	20
Variably dimensional	10
Variably dimensional	20
Wood	10
Wood	20
Zakharov	10
Zakharov	20

Table 2: Results of each strategy of DFPP with no line search over all test problems.

N_s	N_{n1}	N_{n2}	imp
S_n	S_n	S_{2n}	249.01
S_n	S_{2n}	S_n	247.74
S_n	S_{2n}	S_{2n}	234.25
S_n	S_n	S_n	226.75
S_n	$S_{n^2/4}$	S_n	212.73
S_n	$S_{n^2/4}$	S_{2n}	209.35
S_n	$S_{n^2/2}$	S_n	205.58
S_n	S_n	$S_{n^2/4}$	204.31
S_n	S_n	$S_{n^2/2}$	204.19
S_n	$S_{n^2/4}$	$S_{n^2/4}$	202.25
S_n	S_{2n}	$S_{n^2/2}$	202.06
S_{2n}	S_n	S_n	200.75
S_n	S_{2n}	$S_{n^2/4}$	199.24
S_n	$S_{n^2/4}$	$S_{n^2/2}$	198.53
S_n	$S_{n^2/2}$	$S_{n^2/4}$	197.41
S_n	$S_{n^2/2}$	S_{2n}	195.55
S_n	$S_{n^2/2}$	$S_{n^2/2}$	175.70
S_{2n}	S_{2n}	S_n	169.67
S_{2n}	$S_{n^2/4}$	S_n	143.23
S_{2n}	S_{2n}	S_{2n}	143.02
S_{2n}	S_n	S_{2n}	143.02
$S_{n^2/4}$	S_n	S_n	137.59
S_{2n}	$S_{n^2/4}$	S_{2n}	131.18
$S_{n^2/2}$	S_{2n}	S_n	130.88
$S_{n^2/2}$	$S_{n^2/4}$	S_n	130.83
S_{2n}	$S_{n^2/4}$	$S_{n^2/4}$	126.90
S_{2n}	S_{2n}	$S_{n^2/4}$	126.10
S_{2n}	S_n	$S_{n^2/4}$	126.10
$S_{n^2/4}$	S_{2n}	S_n	124.79
$S_{n^2/2}$	S_n	S_n	123.47
S_{2n}	$S_{n^2/2}$	S_n	121.25
$S_{n^2/2}$	S_{2n}	S_{2n}	118.98
$S_{n^2/2}$	S_n	S_{2n}	118.98
S_{2n}	$S_{n^2/4}$	$S_{n^2/2}$	118.09
$S_{n^2/4}$	$S_{n^2/4}$	S_n	117.51
$S_{n^2/4}$	$S_{n^2/4}$	S_{2n}	117.23
S_{2n}	S_{2n}	$S_{n^2/2}$	115.45
S_{2n}	S_n	$S_{n^2/2}$	115.45
$S_{n^2/4}$	S_{2n}	S_{2n}	115.08
$S_{n^2/4}$	S_n	S_{2n}	115.08
$S_{n^2/2}$	$S_{n^2/4}$	S_{2n}	113.97
$S_{n^2/2}$	S_{2n}	$S_{n^2/4}$	113.80
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/4}$	113.80
$S_{n^2/2}$	S_n	$S_{n^2/4}$	113.80
$S_{n^2/2}$	$S_{n^2/2}$	S_n	112.78
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/4}$	110.35
$S_{n^2/2}$	$S_{n^2/2}$	S_{2n}	110.23
$S_{n^2/4}$	S_{2n}	$S_{n^2/4}$	108.27
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/4}$	108.27
$S_{n^2/4}$	S_n	$S_{n^2/4}$	108.27
$S_{n^2/4}$	$S_{n^2/2}$	S_n	105.59
S_{2n}	$S_{n^2/2}$	S_{2n}	103.94
S_{2n}	$S_{n^2/2}$	$S_{n^2/4}$	102.79
$S_{n^2/2}$	S_{2n}	$S_{n^2/2}$	102.45
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/2}$	102.45
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/2}$	102.45
$S_{n^2/2}$	S_n	$S_{n^2/2}$	102.45
$S_{n^2/4}$	S_{2n}	$S_{n^2/2}$	99.11
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/2}$	99.11
$S_{n^2/4}$	S_n	$S_{n^2/2}$	99.11
$S_{n^2/4}$	$S_{n^2/2}$	S_{2n}	94.82
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/4}$	91.17
S_{2n}	$S_{n^2/2}$	$S_{n^2/2}$	91.08
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/2}$	84.84

Table 3: Results of each strategy of DFPP with no line search for low dimensional problems.

N_s	N_{n1}	N_{n2}	imp
S_n	$S_{n^2/2}$	S_n	148.88
S_n	S_n	$S_{n^2/2}$	143.69
S_n	S_n	$S_{n^2/4}$	142.21
S_n	S_n	S_{2n}	140.87
S_n	$S_{n^2/2}$	$S_{n^2/4}$	140.02
S_n	$S_{n^2/4}$	$S_{n^2/2}$	137.87
S_n	$S_{n^2/2}$	S_{2n}	135.96
S_n	$S_{n^2/4}$	$S_{n^2/4}$	135.85
S_n	S_n	S_n	134.79
S_n	S_{2n}	S_n	134.66
S_n	$S_{n^2/4}$	S_n	134.60
S_n	$S_{n^2/4}$	S_{2n}	134.07
S_n	S_{2n}	$S_{n^2/2}$	131.24
S_n	S_{2n}	S_{2n}	131.15
S_n	S_{2n}	$S_{n^2/4}$	127.92
S_n	$S_{n^2/2}$	$S_{n^2/2}$	123.08
S_{2n}	S_n	S_n	109.17
$S_{n^2/4}$	S_n	S_n	102.88
S_{2n}	S_{2n}	S_n	94.27
S_{2n}	$S_{n^2/4}$	S_n	93.35
$S_{n^2/2}$	$S_{n^2/4}$	S_n	93.10
$S_{n^2/2}$	S_{2n}	S_n	93.09
$S_{n^2/4}$	S_{2n}	S_n	90.17
$S_{n^2/2}$	S_n	S_n	86.09
S_{2n}	$S_{n^2/2}$	S_n	85.88
$S_{n^2/4}$	$S_{n^2/4}$	S_n	84.56
$S_{n^2/4}$	$S_{n^2/4}$	S_{2n}	80.79
S_{2n}	$S_{n^2/4}$	S_{2n}	78.81
$S_{n^2/4}$	S_n	S_{2n}	78.71
$S_{n^2/4}$	S_{2n}	S_{2n}	78.71
$S_{n^2/2}$	$S_{n^2/4}$	S_{2n}	78.57
$S_{n^2/2}$	S_n	S_{2n}	78.55
$S_{n^2/2}$	S_{2n}	S_{2n}	78.55
$S_{n^2/2}$	S_n	$S_{n^2/4}$	77.74
$S_{n^2/2}$	S_{2n}	$S_{n^2/4}$	77.74
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/4}$	77.74
$S_{n^2/4}$	$S_{n^2/2}$	S_n	77.65
S_{2n}	S_n	S_{2n}	76.91
S_{2n}	S_{2n}	S_{2n}	76.91
S_{2n}	S_n	$S_{n^2/4}$	76.74
S_{2n}	S_{2n}	$S_{n^2/4}$	76.74
$S_{n^2/4}$	S_n	$S_{n^2/4}$	76.05
$S_{n^2/4}$	S_{2n}	$S_{n^2/4}$	76.05
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/4}$	76.05
$S_{n^2/2}$	$S_{n^2/2}$	S_n	75.57
S_{2n}	$S_{n^2/4}$	$S_{n^2/4}$	75.24
S_{2n}	$S_{n^2/4}$	$S_{n^2/2}$	74.75
$S_{n^2/2}$	$S_{n^2/2}$	S_{2n}	74.71
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/4}$	74.32
S_{2n}	S_n	$S_{n^2/2}$	71.73
S_{2n}	S_{2n}	$S_{n^2/2}$	71.73
S_{2n}	$S_{n^2/2}$	S_{2n}	70.87
$S_{n^2/4}$	S_n	$S_{n^2/2}$	69.64
$S_{n^2/4}$	S_{2n}	$S_{n^2/2}$	69.64
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/2}$	69.64
S_{2n}	$S_{n^2/2}$	$S_{n^2/4}$	68.04
$S_{n^2/4}$	$S_{n^2/2}$	S_{2n}	66.80
$S_{n^2/2}$	S_n	$S_{n^2/2}$	64.68
$S_{n^2/2}$	S_{2n}	$S_{n^2/2}$	64.68
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/2}$	64.68
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/2}$	64.68
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/4}$	63.56
S_{2n}	$S_{n^2/2}$	$S_{n^2/2}$	60.18
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/2}$	55.04

Table 4: Results of each strategy of DFPP with no line search for high dimensional problems.

N_s	N_{n1}	N_{n2}	imp
S_n	S_{2n}	S_n	113.09
S_n	S_n	S_{2n}	108.14
S_n	S_{2n}	S_{2n}	103.09
S_n	S_n	S_n	91.96
S_{2n}	S_n	S_n	91.58
S_n	$S_{n^2/4}$	S_n	78.13
S_{2n}	S_{2n}	S_n	75.40
S_n	$S_{n^2/4}$	S_{2n}	75.28
S_n	S_{2n}	$S_{n^2/4}$	71.32
S_n	S_{2n}	$S_{n^2/2}$	70.82
S_n	$S_{n^2/4}$	$S_{n^2/4}$	66.39
S_{2n}	S_n	S_{2n}	66.11
S_{2n}	S_{2n}	S_{2n}	66.11
S_n	S_n	$S_{n^2/4}$	62.10
S_n	$S_{n^2/4}$	$S_{n^2/2}$	60.66
S_n	S_n	$S_{n^2/2}$	60.50
S_n	$S_{n^2/2}$	S_{2n}	59.59
S_n	$S_{n^2/2}$	$S_{n^2/4}$	57.39
S_n	$S_{n^2/2}$	S_n	56.70
S_n	$S_{n^2/2}$	$S_{n^2/2}$	52.62
S_{2n}	$S_{n^2/4}$	S_{2n}	52.37
S_{2n}	$S_{n^2/4}$	$S_{n^2/4}$	51.66
S_{2n}	$S_{n^2/4}$	S_n	49.88
S_{2n}	S_n	$S_{n^2/4}$	49.36
S_{2n}	S_{2n}	$S_{n^2/4}$	49.36
S_{2n}	S_n	$S_{n^2/2}$	43.72
S_{2n}	S_{2n}	$S_{n^2/2}$	43.72
S_{2n}	$S_{n^2/4}$	$S_{n^2/2}$	43.33
$S_{n^2/2}$	S_n	S_{2n}	40.43
$S_{n^2/2}$	S_{2n}	S_{2n}	40.43
$S_{n^2/2}$	S_{2n}	S_n	37.78
$S_{n^2/2}$	S_n	$S_{n^2/2}$	37.77

N_s	N_{n1}	N_{n2}	imp
$S_{n^2/2}$	S_{2n}	$S_{n^2/2}$	37.77
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/2}$	37.77
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/2}$	37.77
$S_{n^2/2}$	$S_{n^2/4}$	S_n	37.73
$S_{n^2/2}$	S_n	S_n	37.38
$S_{n^2/2}$	$S_{n^2/2}$	S_n	37.21
$S_{n^2/4}$	$S_{n^2/4}$	S_{2n}	36.44
$S_{n^2/4}$	S_n	S_{2n}	36.36
$S_{n^2/4}$	S_{2n}	S_{2n}	36.36
$S_{n^2/2}$	S_n	$S_{n^2/4}$	36.06
$S_{n^2/2}$	S_{2n}	$S_{n^2/4}$	36.06
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/4}$	36.06
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/4}$	36.02
$S_{n^2/2}$	$S_{n^2/2}$	S_{2n}	35.51
$S_{n^2/2}$	$S_{n^2/4}$	S_{2n}	35.40
S_{2n}	$S_{n^2/2}$	S_n	35.37
S_{2n}	$S_{n^2/2}$	$S_{n^2/4}$	34.75
$S_{n^2/4}$	S_n	S_n	34.71
$S_{n^2/4}$	S_{2n}	S_n	34.62
S_{2n}	$S_{n^2/2}$	S_{2n}	33.07
$S_{n^2/4}$	$S_{n^2/4}$	S_n	32.96
$S_{n^2/4}$	S_n	$S_{n^2/4}$	32.22
$S_{n^2/4}$	S_{2n}	$S_{n^2/4}$	32.22
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/4}$	32.22
S_{2n}	$S_{n^2/2}$	$S_{n^2/2}$	30.90
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/2}$	29.80
$S_{n^2/4}$	S_n	$S_{n^2/2}$	29.47
$S_{n^2/4}$	S_{2n}	$S_{n^2/2}$	29.47
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/2}$	29.47
$S_{n^2/4}$	$S_{n^2/2}$	S_{2n}	28.03
$S_{n^2/4}$	$S_{n^2/2}$	S_n	27.93
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/4}$	27.61

Table 5: Results of each strategy of DFPP with line search over all test problems.

N_s	N_{n1}	N_{n2}	imp
S_n	S_n	S_n	377.91
S_n	S_{2n}	S_n	305.56
S_n	S_n	S_{2n}	298.57
S_n	$S_{n^2/2}$	S_n	297.85
S_n	$S_{n^2/4}$	S_n	296.60
S_n	S_n	$S_{n^2/2}$	282.40
S_{2n}	S_n	S_n	282.27
S_n	S_{2n}	$S_{n^2/2}$	279.63
S_n	$S_{n^2/2}$	S_{2n}	279.53
S_n	S_{2n}	S_{2n}	278.18
S_n	$S_{n^2/4}$	$S_{n^2/2}$	271.84
S_{2n}	S_{2n}	S_n	267.72
S_n	S_n	$S_{n^2/4}$	266.60
$S_{n^2/2}$	S_n	S_n	263.12
S_n	$S_{n^2/2}$	$S_{n^2/2}$	258.12
$S_{n^2/2}$	S_{2n}	S_n	256.75
$S_{n^2/2}$	$S_{n^2/4}$	S_n	254.52
S_{2n}	$S_{n^2/4}$	S_n	251.04
S_n	$S_{n^2/4}$	S_{2n}	250.89
$S_{n^2/2}$	$S_{n^2/2}$	S_n	246.53
S_{2n}	S_{2n}	S_{2n}	245.19
S_{2n}	S_n	S_{2n}	245.19
$S_{n^2/4}$	S_{2n}	S_n	242.61
$S_{n^2/4}$	S_n	S_n	242.11
$S_{n^2/2}$	S_{2n}	S_{2n}	241.41
$S_{n^2/2}$	S_n	S_{2n}	241.41
$S_{n^2/2}$	$S_{n^2/4}$	S_{2n}	239.60
S_{2n}	S_{2n}	$S_{n^2/2}$	238.97
S_{2n}	S_n	$S_{n^2/2}$	238.97
$S_{n^2/2}$	$S_{n^2/2}$	S_{2n}	238.53
S_n	$S_{n^2/2}$	$S_{n^2/4}$	237.97
S_n	$S_{n^2/4}$	$S_{n^2/4}$	237.64
S_{2n}	$S_{n^2/2}$	S_n	237.10
$S_{n^2/4}$	$S_{n^2/4}$	S_n	234.47
$S_{n^2/2}$	S_{2n}	$S_{n^2/4}$	233.97
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/4}$	233.97
$S_{n^2/2}$	S_n	$S_{n^2/4}$	233.97
S_n	S_{2n}	$S_{n^2/4}$	233.34
S_{2n}	$S_{n^2/4}$	$S_{n^2/2}$	232.09
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/4}$	231.28
$S_{n^2/4}$	S_{2n}	$S_{n^2/2}$	229.72
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/2}$	229.72
$S_{n^2/4}$	S_n	$S_{n^2/2}$	229.72
$S_{n^2/2}$	S_{2n}	$S_{n^2/2}$	228.39
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/2}$	228.39
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/2}$	228.39
$S_{n^2/2}$	S_n	$S_{n^2/2}$	228.39
S_{2n}	$S_{n^2/2}$	$S_{n^2/2}$	227.36
$S_{n^2/4}$	$S_{n^2/2}$	S_n	226.71
$S_{n^2/4}$	S_{2n}	S_{2n}	226.51
$S_{n^2/4}$	S_n	S_{2n}	226.51
S_{2n}	$S_{n^2/4}$	S_{2n}	225.04
$S_{n^2/4}$	$S_{n^2/4}$	S_{2n}	224.73
$S_{n^2/4}$	$S_{n^2/2}$	S_{2n}	223.34
S_{2n}	S_{2n}	$S_{n^2/4}$	223.05
S_{2n}	S_n	$S_{n^2/4}$	223.05
S_{2n}	$S_{n^2/2}$	S_{2n}	222.63
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/2}$	219.03
S_{2n}	$S_{n^2/4}$	$S_{n^2/4}$	214.59
S_{2n}	$S_{n^2/2}$	$S_{n^2/4}$	214.35
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/4}$	210.46
$S_{n^2/4}$	S_{2n}	$S_{n^2/4}$	208.80
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/4}$	208.80
$S_{n^2/4}$	S_n	$S_{n^2/4}$	208.80

Table 6: Results of each strategy of DFPP with line search for low dimensional problems.

N_s	N_{n1}	N_{n2}	imp
S_n	S_n	S_n	153.38
S_n	S_n	$S_{n^2/2}$	149.57
S_n	S_{2n}	$S_{n^2/2}$	147.67
S_n	$S_{n^2/4}$	$S_{n^2/2}$	146.20
S_n	S_n	$S_{n^2/4}$	142.24
S_n	$S_{n^2/2}$	S_n	141.89
$S_{n^2/2}$	S_n	S_n	141.23
S_{2n}	$S_{n^2/4}$	$S_{n^2/2}$	141.03
$S_{n^2/4}$	S_n	$S_{n^2/2}$	140.20
$S_{n^2/4}$	S_{2n}	$S_{n^2/2}$	140.20
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/2}$	140.20
S_n	$S_{n^2/2}$	$S_{n^2/2}$	138.87
S_{2n}	S_n	$S_{n^2/2}$	138.75
S_{2n}	S_{2n}	$S_{n^2/2}$	138.75
S_n	S_n	S_{2n}	138.48
S_n	$S_{n^2/2}$	S_{2n}	138.35
S_n	$S_{n^2/2}$	$S_{n^2/4}$	137.88
S_{2n}	S_n	S_n	136.62
$S_{n^2/4}$	S_n	S_n	136.49
$S_{n^2/2}$	$S_{n^2/4}$	S_n	136.46
$S_{n^2/2}$	S_{2n}	S_n	136.32
S_{2n}	S_{2n}	S_n	135.27
$S_{n^2/4}$	$S_{n^2/4}$	S_n	134.27
S_{2n}	$S_{n^2/4}$	S_n	133.99
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/2}$	133.51
$S_{n^2/4}$	S_{2n}	S_n	133.43
S_{2n}	$S_{n^2/2}$	$S_{n^2/2}$	130.36
$S_{n^2/2}$	S_n	S_{2n}	130.11
$S_{n^2/2}$	S_{2n}	S_{2n}	130.11
$S_{n^2/2}$	S_n	$S_{n^2/4}$	130.05
$S_{n^2/2}$	S_{2n}	$S_{n^2/4}$	130.05
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/4}$	130.05

N_s	N_{n1}	N_{n2}	imp
$S_{n^2/2}$	$S_{n^2/4}$	S_{2n}	128.62
$S_{n^2/2}$	$S_{n^2/2}$	S_n	128.58
S_n	$S_{n^2/4}$	S_{2n}	128.43
$S_{n^2/2}$	S_n	$S_{n^2/2}$	127.88
$S_{n^2/2}$	S_{2n}	$S_{n^2/2}$	127.88
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/2}$	127.88
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/2}$	127.88
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/4}$	127.86
S_n	S_{2n}	$S_{n^2/4}$	126.56
S_n	$S_{n^2/4}$	$S_{n^2/4}$	125.88
$S_{n^2/4}$	$S_{n^2/4}$	S_{2n}	125.31
$S_{n^2/4}$	S_n	S_{2n}	125.21
$S_{n^2/4}$	S_{2n}	S_{2n}	125.21
$S_{n^2/2}$	$S_{n^2/2}$	S_{2n}	125.20
S_n	S_{2n}	S_{2n}	124.94
$S_{n^2/4}$	S_n	$S_{n^2/4}$	124.50
$S_{n^2/4}$	S_{2n}	$S_{n^2/4}$	124.50
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/4}$	124.50
$S_{n^2/4}$	$S_{n^2/2}$	S_n	124.07
S_{2n}	$S_{n^2/2}$	S_n	123.75
S_{2n}	S_n	$S_{n^2/4}$	123.05
S_{2n}	S_{2n}	$S_{n^2/4}$	123.05
S_{2n}	$S_{n^2/4}$	$S_{n^2/4}$	123.04
S_{2n}	$S_{n^2/4}$	S_{2n}	121.54
S_n	S_{2n}	S_n	121.36
$S_{n^2/4}$	$S_{n^2/2}$	S_{2n}	120.69
S_n	$S_{n^2/4}$	S_n	120.03
S_{2n}	$S_{n^2/2}$	$S_{n^2/4}$	119.10
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/4}$	118.97
S_{2n}	S_n	S_{2n}	118.19
S_{2n}	S_{2n}	S_{2n}	118.19
S_{2n}	$S_{n^2/2}$	S_{2n}	117.63

Table 7: Results of each strategy of `DFPP` with line search for high dimensional problems.

N_s	N_{n1}	N_{n2}	`imp`
S_n	S_n	S_n	224.53
S_n	S_{2n}	S_n	184.19
S_n	$S_{n^2/4}$	S_n	176.56
S_n	S_n	S_{2n}	160.09
S_n	$S_{n^2/2}$	S_n	155.96
S_n	S_{2n}	S_{2n}	153.24
S_{2n}	S_n	S_n	145.65
S_n	$S_{n^2/2}$	S_{2n}	141.18
S_n	S_n	$S_{n^2/2}$	132.83
S_{2n}	S_{2n}	S_n	132.45
S_n	S_{2n}	$S_{n^2/2}$	131.96
S_{2n}	S_n	S_{2n}	127.00
S_{2n}	S_{2n}	S_{2n}	127.00
S_n	$S_{n^2/4}$	$S_{n^2/2}$	125.63
S_n	S_n	$S_{n^2/4}$	124.36
S_n	$S_{n^2/4}$	S_{2n}	122.46
$S_{n^2/2}$	S_n	S_n	121.89
$S_{n^2/2}$	S_{2n}	S_n	120.43
S_n	$S_{n^2/2}$	$S_{n^2/2}$	119.25
$S_{n^2/2}$	$S_{n^2/4}$	S_n	118.06
$S_{n^2/2}$	$S_{n^2/2}$	S_n	117.95
S_{2n}	$S_{n^2/4}$	S_n	117.05
S_{2n}	$S_{n^2/2}$	S_n	113.35
$S_{n^2/2}$	$S_{n^2/2}$	S_{2n}	113.32
S_n	$S_{n^2/4}$	$S_{n^2/4}$	111.75
$S_{n^2/2}$	S_n	S_{2n}	111.30
$S_{n^2/2}$	S_{2n}	S_{2n}	111.30
$S_{n^2/2}$	$S_{n^2/4}$	S_{2n}	110.99
$S_{n^2/4}$	S_{2n}	S_n	109.18
S_n	S_{2n}	$S_{n^2/4}$	106.78
$S_{n^2/4}$	S_n	S_n	105.62
S_{2n}	$S_{n^2/2}$	S_{2n}	105.00

N_s	N_{n1}	N_{n2}	`imp`
$S_{n^2/2}$	S_n	$S_{n^2/4}$	103.92
$S_{n^2/2}$	S_{2n}	$S_{n^2/4}$	103.92
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/4}$	103.92
S_{2n}	$S_{n^2/4}$	S_{2n}	103.51
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/4}$	103.43
$S_{n^2/4}$	$S_{n^2/2}$	S_{2n}	102.65
$S_{n^2/4}$	$S_{n^2/2}$	S_n	102.65
$S_{n^2/4}$	S_n	S_{2n}	101.30
$S_{n^2/4}$	S_{2n}	S_{2n}	101.30
$S_{n^2/2}$	S_n	$S_{n^2/2}$	100.51
$S_{n^2/2}$	S_{2n}	$S_{n^2/2}$	100.51
$S_{n^2/2}$	$S_{n^2/4}$	$S_{n^2/2}$	100.51
$S_{n^2/2}$	$S_{n^2/2}$	$S_{n^2/2}$	100.51
S_{2n}	S_n	$S_{n^2/2}$	100.23
S_{2n}	S_{2n}	$S_{n^2/2}$	100.23
$S_{n^2/4}$	$S_{n^2/4}$	S_n	100.19
S_n	$S_{n^2/2}$	$S_{n^2/4}$	100.09
S_{2n}	S_n	$S_{n^2/4}$	100.00
S_{2n}	S_{2n}	$S_{n^2/4}$	100.00
$S_{n^2/4}$	$S_{n^2/4}$	S_{2n}	99.42
S_{2n}	$S_{n^2/2}$	$S_{n^2/2}$	97.00
S_{2n}	$S_{n^2/2}$	$S_{n^2/4}$	95.24
S_{2n}	$S_{n^2/4}$	$S_{n^2/4}$	91.55
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/4}$	91.49
S_{2n}	$S_{n^2/4}$	$S_{n^2/2}$	91.06
$S_{n^2/4}$	S_n	$S_{n^2/2}$	89.52
$S_{n^2/4}$	S_{2n}	$S_{n^2/2}$	89.52
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/2}$	89.52
$S_{n^2/4}$	$S_{n^2/2}$	$S_{n^2/2}$	85.52
$S_{n^2/4}$	S_n	$S_{n^2/4}$	84.30
$S_{n^2/4}$	S_{2n}	$S_{n^2/4}$	84.30
$S_{n^2/4}$	$S_{n^2/4}$	$S_{n^2/4}$	84.30