# Adaptive Discontinuous Galerkin Finite Element Methods for Second and Fourth Order Elliptic Partial Differential Equations

***Dissertation Defense***

Michael A. Saum

msaum@math.utk.edu

Department of Mathematics

University of Tennessee, Knoxville

# Overview

- Contributions
- Brief Overview of the DG Method
- A Posteriori Error Estimation
- Adaptive Methods
- Data Structures
- Performance Monitoring and Optimization
- Blocking for Cache
- Linear Solvers
- Results
- Future Work

# Contributions

Developed and implemented working adaptive versions of DG-FEM for second and fourth order elliptic PDEs.

- `Ell2` $\sim$ 13,000 lines of `C` code.
- `Ell4` $\sim$ 15,000 lines of `C` code.
- Modular design allowed for $\sim$ 8,000 lines of `Ell2` code to be used in `Ell4` without change.
- Implemented Linear Solvers: CG, MG, PCG/MG.
- Utilized existing state of the art software where possible including `ATLAS`, `Clapack`, `Triangle`, `PAPI`, `METIS`, and `MeshTV/SILO`

# Contribs, contd.

- Implemented *Cache Blocking* for Gauss-Seidel utilizing ideas of Douglas et al. (2000).
- Designed and implemented data structures which work well within an adaptive DG-FEM scientific computing environment.
- Extended prior results of Karakashian and Pascal (2003, 2004) regarding DG formulation of second order elliptic and biharmonic PDEs for Arnold and Baker formulations.

# Contribs, contd.

- Obtained explicit formulations of local problem right-hand sides for Arnold and Baker formulations of the biharmonic equation.
- Source Code will be packaged and made available in the future.
- `Ell2` and `Ell4` provide an excellent platform for investigating numerical characteristics of adaptive DG-FEM PDE models.

# DG Overview

- The Discontinuous Galerkin (DG) Finite Element Method (FEM) is a variant of the Standard (Continuous) Galerkin (SG) FEM.
- SG-FEM requires continuity of the solution along element interfaces (edges).
- DG-FEM does not require continuity of the solution along edges.
- DG methods have more degrees of freedom (unknowns) to solve for than SG methods.

# DG Advantages

- DG methods have what can be considered to be a number of advantages over SG methods:
  - Global stiffness matrix contains a very nice block structure, our formulation produces a symmetric, positive definite linear system to be solved.
  - Regular triangle refinement produces a *Natural Hierarchy* allowing for multilevel methods to be integrated into solvers.
- DG methods can support high order local approximations that can vary nonuniformly over the mesh.

# Ell2 – Model Problem

Let $\Omega \subset \mathbb{R}^d, d = 2, 3$ be a bounded open polyhedral domain with Lipshitz continuous boundary.

$$\begin{cases} -\Delta u = f & \text{in } \Omega \\ u = g_D & \text{on } \Gamma_D \\ \nabla u \cdot n = g_N & \text{on } \Gamma_N \end{cases} \qquad \text{(MP)}$$

where $\partial\Omega := \Gamma = \Gamma_D \cup \Gamma_N$ and $n$ is the unit normal vector exterior to $\Omega$. We also assume that $\mu_{d-1}(\Gamma_D) > 0$, $f \in L^2(\Omega)$, $g_N \in L^2(\Gamma_N)$.

# Notation

- Let $\mathscr{T}_h = \{K_i : i = 1, 2, \ldots, m_h\}$ be a family of star-like partitions of $\Omega$ parameterized by $0 < h \leq 1$.
- The elements of $\mathscr{T}_h$ satisfy the minimal angle condition.
- $\mathscr{T}_h$ is locally quasi-uniform.
- $\mathscr{E}^I = \{e = \partial K_j \cap \partial K_l : \mu_{d-1}(\partial K_j \cap \partial K_l) > 0\}$
- $\mathscr{E}^B = \{e = \partial K_j \cap \partial \Omega : \mu_{d-1}(\partial K_j \cap \partial \Omega) > 0\}$
- $\forall e \in \mathscr{E}^B$, either $e \subset \Gamma_D$ or $e \subset \Gamma_N$ and $\mathscr{E} = \mathscr{E}^I \cup \mathscr{E}^B$, where $\mathscr{E}^B = \mathscr{E}^B_D \cup \mathscr{E}^B_N$ and $\mathscr{E}^B_D \cap \mathscr{E}^B_N = \emptyset$.
- If $e \in \mathscr{E}^I$, then $e = \partial K^+ \cap \partial K^-$ for $K^+, K^- \in \mathscr{T}_h$.
- If $e \in \mathscr{E}^B$, then $e = \partial K^+ \cap \partial \Omega \equiv \partial K \cap \partial \Omega$.
- $n^+$ is the unit normal to $e$ that points outward from $K^+$.
- On $\mathscr{T}_h$, for $r \geq 2$, defi ne the energy space $E_h$ and fi nite element space $V_h^r$ by

$$E_h = \prod_{K \in \mathscr{T}_h} H^2(K), \quad V_h^r = \prod_{K \in \mathscr{T}_h} P_k(K)$$

where $P_k(K)$ denotes the space of polynomials of total degree $r - 1 \equiv k \geq 1$.

# Weak Formulation

- First obtain weak formulation by multiplying (MP) by $v \in V_h^r$ and integrating over $\Omega$:

$$-\int_\Omega (\Delta u) v \, dx = \int_\Omega f v \, dx$$

- Now decompose integrals into element contributions and integrate by parts:

$$\sum_{K \in \mathcal{T}_h} -\int_K (\Delta u) v \, dx = \sum_{K \in \mathcal{T}_h} \int_K f v \, dx$$

$$\sum_{K \in \mathcal{T}_h} \int_K \nabla u \cdot \nabla v \, dx - \sum_{K \in \mathcal{T}_h} \int_{\partial K} \frac{\partial u}{\partial n} v \, ds = \sum_{K \in \mathcal{T}_h} \int_K f v \, dx$$

# Weak Formulation, contd.

- Splitting Edge integrals:

$$\sum_{K\in\mathcal{T}_h}\left\langle\frac{\partial u}{\partial n},v\right\rangle_{\partial K}=\sum_{e\in\Gamma_D}\left\langle\frac{\partial u}{\partial n},v\right\rangle_e+\sum_{e\in\Gamma_N}\left\langle\frac{\partial u}{\partial n},v\right\rangle_e$$
$$+\sum_{e\in\mathcal{E}^I}\left(\left\langle\frac{\partial u^+}{\partial n^+},v\right\rangle_e+\left\langle\frac{\partial u^-}{\partial n^-},v\right\rangle_e\right)$$

- Resulting in:

$$\sum_{K\in\mathcal{T}_h}(\nabla u,\nabla v)_K-\left\langle\frac{\partial u}{\partial n},v\right\rangle_{\Gamma_D}-\sum_{e\in\mathcal{E}^I}\left(\left\langle\frac{\partial u^+}{\partial n^+},v\right\rangle_e-\left\langle\frac{\partial u^-}{\partial n^+},v\right\rangle_e\right)$$
$$=\sum_{K\in\mathcal{T}_h}(f,v)_K+\langle g_N,v\rangle_{\Gamma_N}$$

# Weak Formulation, contd.

- One can treat the above internal edge integrals using the following identities:
  - D. Arnold (Arnold, 1982): $ac - bd = \frac{1}{2}(a+b)(c-d) + \frac{1}{2}(a-b)(c+d)$.
  - G. Baker (Baker, 1977): $ac - bd = a(c-d) + (a-b)d$.
- Define
  - $B(u,v) := \sum_{K \in \mathcal{T}_h} (\nabla u, \nabla v)_K$
  - $F(v) := \sum_{K \in \mathcal{T}_h} (f,v)_K + \langle g_N, v \rangle_{\Gamma_N}$
  - $J(u,v) := \left\langle \frac{\partial u}{\partial n}, v \right\rangle_{\Gamma_D} + \sum_{e \in \mathcal{E}^I} \left\langle \left\{ \frac{\partial u}{\partial n} \right\}, [v] \right\rangle_e$
  - where $\left\{ \frac{\partial u}{\partial n} \right\}\Big|_e = \frac{1}{2}\left( \frac{\partial u^+}{\partial n} + \frac{\partial u^-}{\partial n} \right)\Big|_e$ (Arnold) and,
  - $\left\{ \frac{\partial u}{\partial n} \right\}\Big|_e = \frac{\partial u^+}{\partial n}\Big|_e$ (Baker) , and
  - $[v]\big|_e = \left(v^+ - v^-\right)\Big|_e$.

# SIPG Formulation

- Leads to a weak formulation of (MP): Find $u \in H^2(\Omega)$ such that

$$B(u,v) - J(u,v) = F(v) \quad \forall v \in E_h$$

- Symmetric Interior Penalty Formulation (SIPG) involves modifications:
  - Symmetrization:

$$B(u,v) - J(u,v) - J(v,u) = F(v) - \left\langle \frac{\partial v}{\partial n}, g_D \right\rangle_{\Gamma_D}$$

  - Note that $\langle \cdot, [u] \rangle_{e \in \mathcal{E}^I} = 0$ for $u \in H^1(\Omega) \cap E_h$.

# SIPG Formulation, contd.

- Penalization of *jump* terms:
  - Let $\gamma > 0$ be a penalization parameter
  - Let $J^\gamma(u,v) := \sum_{e \in \mathcal{E}^I} \left\langle \gamma h_e^{-1} [u], [v] \right\rangle_e + \left\langle \gamma h_e^{-1} u, v \right\rangle_{\Gamma_D}$
- SIPG Formulation: Find $u \in H^1 \cap E_h$ such that

$$B(u,v) - J(u,v) - J(v,u) + J^\gamma(u,v)$$

$$= F(v) - \left\langle \frac{\partial v}{\partial n}, g_D \right\rangle_{\Gamma_D} + \left\langle \gamma h_e^{-1} g_D, v \right\rangle_{\Gamma_D} \quad \forall v \in E_h$$

# Ell2 – DG FEM Formulation

Find $u_h^\gamma \in V_h^r$ such that

$$a_h^\gamma\left(u_h^\gamma, v\right) = F_h^\gamma(v), \qquad \forall v \in V_h^r \tag{1}$$

where

$$a_h^\gamma\left(u_h^\gamma, v\right) = \sum_{K\in\mathscr{T}_h} (\nabla u_h^\gamma, \nabla v)_K$$

$$- \sum_{e\in\mathscr{E}^I\cup\mathscr{E}_D^B} \left( \left\langle \left\{\partial_n u_h^\gamma\right\}, [v] \right\rangle_e + \left\langle \left\{\partial_n v\right\}, \left[u_h^\gamma\right] \right\rangle_e - \gamma h_e^{-1} \left\langle [u_h^\gamma], [v] \right\rangle_e \right) \tag{2}$$

and

$$F_h^\gamma(v) = \sum_{K\in\mathscr{T}_h} (f, v)_K - \left\langle g_D, \partial_n v - \gamma h_e^{-1} v \right\rangle_{\Gamma_D} + \left\langle g_N, v \right\rangle_{\Gamma_N} \tag{3}$$

# Ell2 – Energy Norm

- The bilinear form $a_h^\gamma(\cdot,\cdot)$ induces the following norm on $E_h$:

$$\|v\|_{1,h} = \left( \sum_{K\in\mathscr{T}_h} \|\nabla v\|_{0,K}^2 + \sum_{e\in\mathscr{E}^I\cup\mathscr{E}_D^B} \left( h_e^{-1} |[v]|_{0,e}^2 + h_e |\{\partial_n v\}|_{0,e}^2 \right) \right)^{1/2}$$

- Note that $a_h^\gamma(\cdot,\cdot)$ is symmetric, coercive for $\gamma > \gamma_0 > 0$ for $\gamma_0$ large enough.
- Note also that $\gamma = \gamma(r)$. For second order elliptic problems, it is common to take $\gamma(r) = \gamma_c (r-1)^2$, and use the condition $\gamma_c > \gamma_0$ for $\gamma_0$ large enough.

# Ell4 – Model Problem

The fourth order elliptic model problem under consideration is:

$$\begin{cases} \Delta^2 u = f & \text{in } \Omega \\ u = g_D & \text{on } \Gamma \\ \nabla u \cdot n = g_N & \text{on } \Gamma \end{cases} \qquad \text{(MP)}$$

where $\Omega \subset \mathbb{R}^d$, $d = 2, 3$ and $\partial\Omega = \Gamma$ with $n$ being the unit outward normal vector to $\Gamma$.

# Ell4 – Energy Spaces

Let the *energy spaces* $E_h$ be defined as

$$E_h = \prod_{K \in \mathcal{T}_h} H^4(K)$$

and the *finite element spaces* $V_h^r$ be defined as

$$V_h^r = \prod_{K \in \mathcal{T}_h} P_{r-1}(K)$$

where $P_{r-1}(K)$ denotes the space of polynomials of total degree $r-1$ on $K$. Note that $V_h^r \subset E_h \subset L^2(\Omega)$, but $V_h^r \not\subset H^2(\Omega)$ and $V_h^r \not\subset H^1(\Omega)$.

# Ell4 – DG FEM Formulation

Find $u_h^\gamma \in V_h^r$ such that

$$a_h^\gamma\left(u_h^\gamma, v\right) = F_h^\gamma(v), \qquad \forall v \in V_h^r \tag{4}$$

where

$$\begin{aligned} a_h^\gamma\left(u_h^\gamma, v\right) = & \sum_{K\in\mathscr{T}_h} \left(\Delta u_h^\gamma, \Delta v\right)_K \\ & + \sum_{e\in\mathscr{E}} \Big( \left\langle \{\partial_n(\Delta v)\}, [u_h^\gamma] \right\rangle_e - \left\langle \{\Delta v\}, \left[\partial_n u_h^\gamma\right] \right\rangle_e + \left\langle \left\{\partial_n(\Delta u_h^\gamma)\right\}, [v] \right\rangle_e - \left\langle \left\{\Delta u_h^\gamma\right\}, [\partial_n v] \right\rangle_e \\ & \qquad + \gamma h_e^{-1} \left\langle \left[\partial_n u_h^\gamma\right], [\partial_n v] \right\rangle_e + \gamma h_e^{-3} \left\langle [u_h^\gamma], [v] \right\rangle_e \Big) \end{aligned} \tag{5}$$

and

$$F_h^\gamma(v) = \sum_{K\in\mathscr{T}_h} (f, v)_K + \sum_{e\in\Gamma} \left( \left\langle g_D, \partial_n(\Delta v) + \gamma h_e^{-3} v \right\rangle_e + \left\langle g_N, \gamma h_e^{-1} \partial_n v - \Delta v \right\rangle_e \right) \tag{6}$$

# Ell4 – Energy Norm

The bilinear form $a_h^\gamma(\cdot,\cdot)$ induces the following norms on $E_h$:

$$\|v\|_{2,h} = \left( \sum_{K\in\mathscr{T}_h} \|\Delta v\|_{0,K}^2 + \sum_{e\in\mathscr{E}} \left( h_e^{-3} |[v]|_{0,e}^2 + h_e^{-1} |[\partial_n v]|_{0,e}^2 + h_e |\{\Delta v\}|_{0,e}^2 + h_e^3 |\{\partial_n(\Delta v)\}|_{0,e}^2 \right) \right)^{1/2} \tag{7}$$

and

$$\|v\|_{1,h} = \left( \sum_{K\in\mathscr{T}_h} \|\nabla v\|_{0,K}^2 + \sum_{e\in\mathscr{E}} \left( h_e^{-1} |[v]|_{0,e}^2 + h_e |\{\partial_n v\}|_{0,e}^2 \right) \right)^{1/2} \tag{8}$$

## A Posteriori Error Estimation

- A posteriori error estimates rely on computed solutions to provide indicators into regions of the domain where the solution can be improved.
- Identifying the appropiate combination of the computed solution, residuals, and boundary data to produce residual based sharp a posteriori error indicators is the key challenge.
- Many different types of estimators exist. For an excellent summary of a posteriori error estimation, refer to Verfürth (1995), Babuşka and Strouboulis (2001).

# A Posteriori Error Est., contd

The following theorem stated without proof (Karakashian and Pascal, 2004) provides a residual based a posteriori estimator for our second order elliptic problem.

**Theorem.** *Let* $e = u - u_h^\gamma$. *Then*

$$\sum_{K\in\mathcal{T}_h} \|\nabla e\|_K^2 \le c\Big( \sum_{K\in\mathcal{T}_h} h_K^2 \| f + \Delta u_h^\gamma \|_K^2 + \sum_{e\in\mathcal{E}^I} h_e |[\partial_n u_h^\gamma]|_e^2 + \sum_{e\in\mathcal{E}_N^B} h_e |g_N - \partial_n u_h^\gamma|_e^2 + \gamma^2 \sum_{e\in\mathcal{E}^I} h_e^{-1} |[u_h^\gamma]|_e^2 + \gamma^2 \sum_{e\in\mathcal{E}_D^B} h_e^{-1} |g_D - u_h^\gamma|_e^2 \Big)$$

*Note: The presence of* $\gamma^2$ *is necessary, compare with only* $\gamma$ *in the bilinear form.*

# Adaptive Methods

- Uniform refinement is overkill for some problems. For example, near a singular point the solution varies quite rapidly, but far away from a singular point the solution may not vary much at all.
- An *Adaptive Iteration* consists of a Solve, Estimate, Mark, Refine, Coarsen sequence, usually abbreviated to *SER* or Solve-Estimate-Refine.
- Adaptive iterations terminate when the desired estimator tolerance is achieved, i.e., the adaptive scheme is convergent.

# Adaptive Methods, contd.

- Very important to any adaptive scheme is the marking strategy used to identify candidates for refinement and coarsening.
- We utilize a modification of the marking strategy employed by Dörfler (1996), whose scheme was proven to be convergent.
- In a nutshell, after computing *local* estimators $\eta_K, \forall K \in \mathscr{T}_h$, sorting in decreasing order, we mark until we reach a certain fraction $\theta \in (0,1)$ of the global estimator total $\theta$.

# Dörfler Marking Strategy

**Require:** Fix $\theta \in (0,1)$
**Require:** Fix $\nu \in (0,1)$, small
  $\mathscr{S} = \emptyset$
  $s = 0$
  $\tau = 1$
  **while** $s < \theta^2 \eta_{\mathscr{T}}^2$ **do**
    $\tau = \tau - \nu$
    **for all** $K \in \mathscr{T}_h$ **do**
      **if** $K$ is not marked **then**
        **if** $\eta_K > \tau \eta_{\max}$ **then**
          Mark $K$, $\mathscr{S} = \mathscr{S} + K$
          $s = s + \eta_K^2$
        **end if**
      **end if**
    **end for**
  **end while**

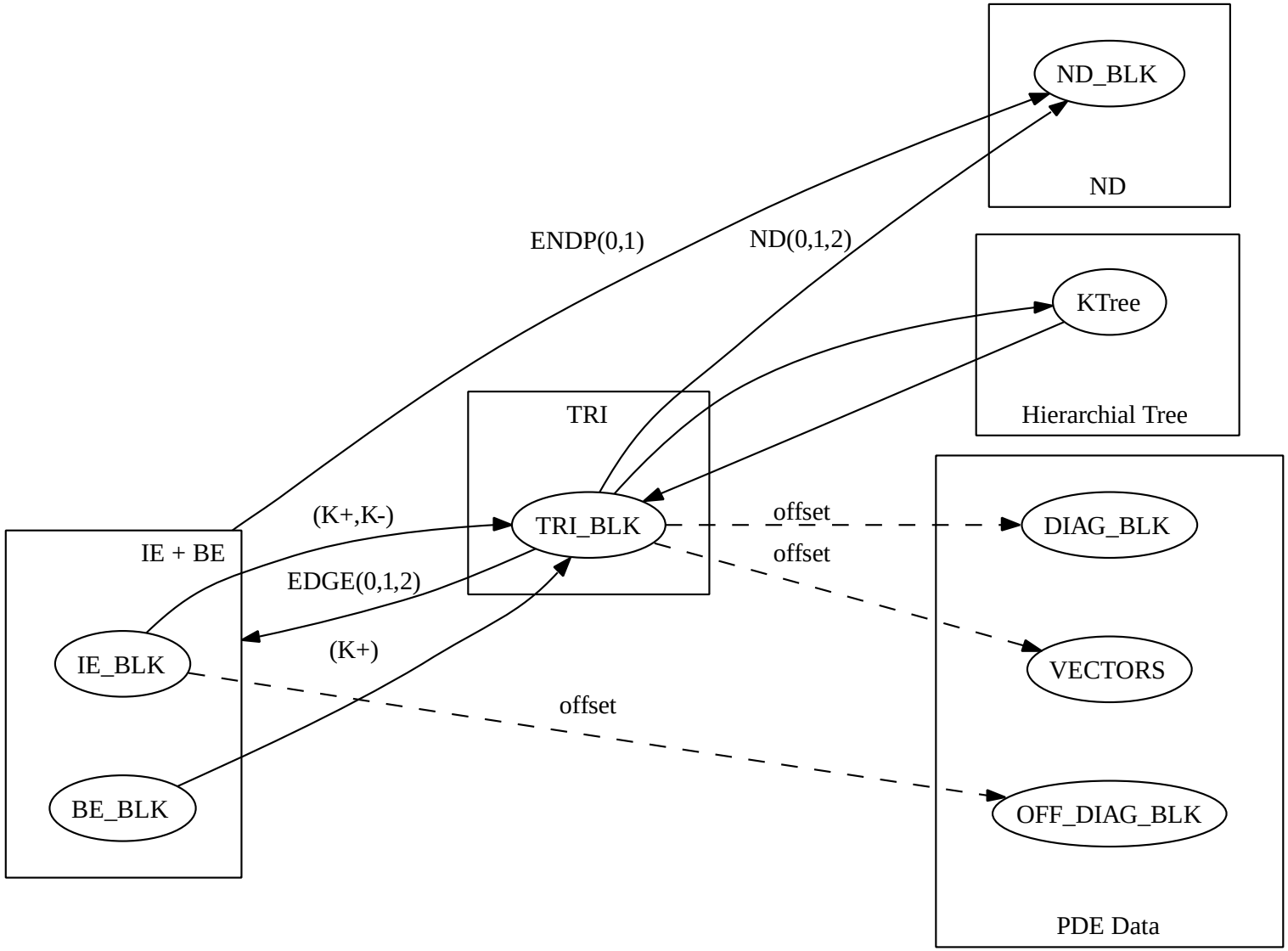
# Adaptive Methods, contd.

- Another strategy based on recent work by O. Karakashian (Karakashian and Pascal, 2006) utilizes a combination of triangle marking and edge marking (which induces triangle marking) for refinement which produces a convergent adaptive algorithm.
- It is common to use a fixed value for $\theta$, noting that if $\theta \approx 1$ then most triangles will be chosen to be refined while if $\theta \approx 0$ then very few triangles will be selected for refinement.
- We have started investigation into choosing a variable $\theta$ which has shown to work in practice, the theory is still in the research phase.

# Data Structures

- `C` and `FORTRAN` concepts used for memory utilization (the best of both worlds).
- Geometric data objects include TRIANGLE, EDGE, and NODE.
- Objects stored in one long array for each data object type and managed via doubly linked list structures.
- Pointers are used to identify relations between objects.
- Hierarchial relations are stored in a 4-ary tree structure rooted in the initial mesh.
- PDE data (vectors, stiffness matrix blocks) are stored separately from geometric data but follow the order of storage of geometric data objects.

# Data Structure Relations



# Ell2 – Test Problem f3

Test Problem - f3 Domain $\Omega$: Figure 1

$$\begin{cases} -\Delta u = 2\pi^2 \sin(\pi x)\sin(\pi y) & \text{in } \Omega \\ u = 0 & \text{on } \Gamma_D \end{cases}$$

Exact solution: $u = \sin(\pi x)\sin(\pi y)$.

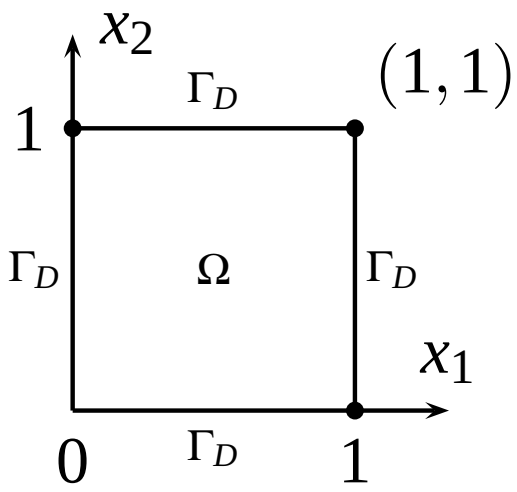



Figure 1: Square Domain

# Ell2 – Test Problem f4

Test Problem - f4 Domain $\Omega$: Figure 1

$$\begin{cases} -\Delta u = 128\pi^2 \sin(8\pi x)\sin(8\pi y) & \text{in } \Omega \\ u = 0 & \text{on } \Gamma_D \end{cases}$$

Exact solution: $u = \sin(8\pi x)\sin(8\pi y)$.

# Ell2 – Test Problem f6

Test Problem - f6 Domain $\Omega$: Figure 2

$$\begin{cases} \Delta u = 0 & \text{in } \Omega \\ u = r^{2/3}\sin(2/3\theta) & \text{on } \Gamma_D \end{cases}$$

Exact solution: $u = r^{2/3}\sin(2/3\theta)$.

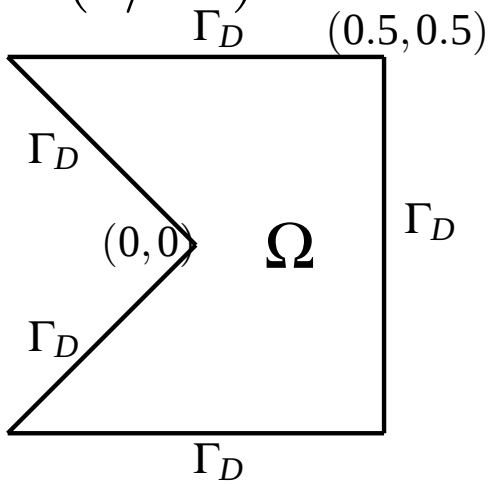



Figure 2: Notch Domain

# Peformance Optimization

Choice of compilers and compiler optimization flags can affect program performance. The following compilers and optimization levels are compared in Figures 3–4:

- NoOpt: `gcc -O0` - No Optimization
- O2Opt: `gcc -O2` - Medium Optimization
- FullOpt: `gcc` - Aggressive Optimization
- InOpt: `icc` - Aggressive Optimization (Intel)

# Perf. Opt., contd.

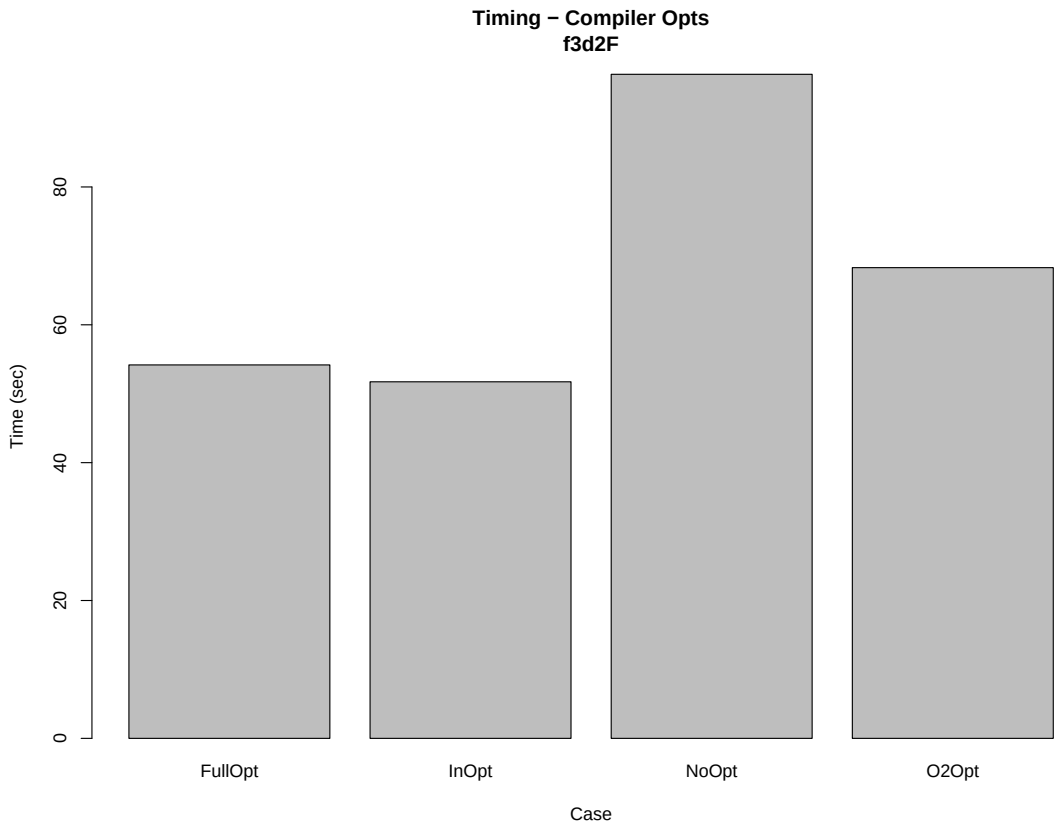



Figure 3: Performance Opt. Time (s), f3, $r = 3$, Uniform, 393216 dof

# Perf. Opt., contd.

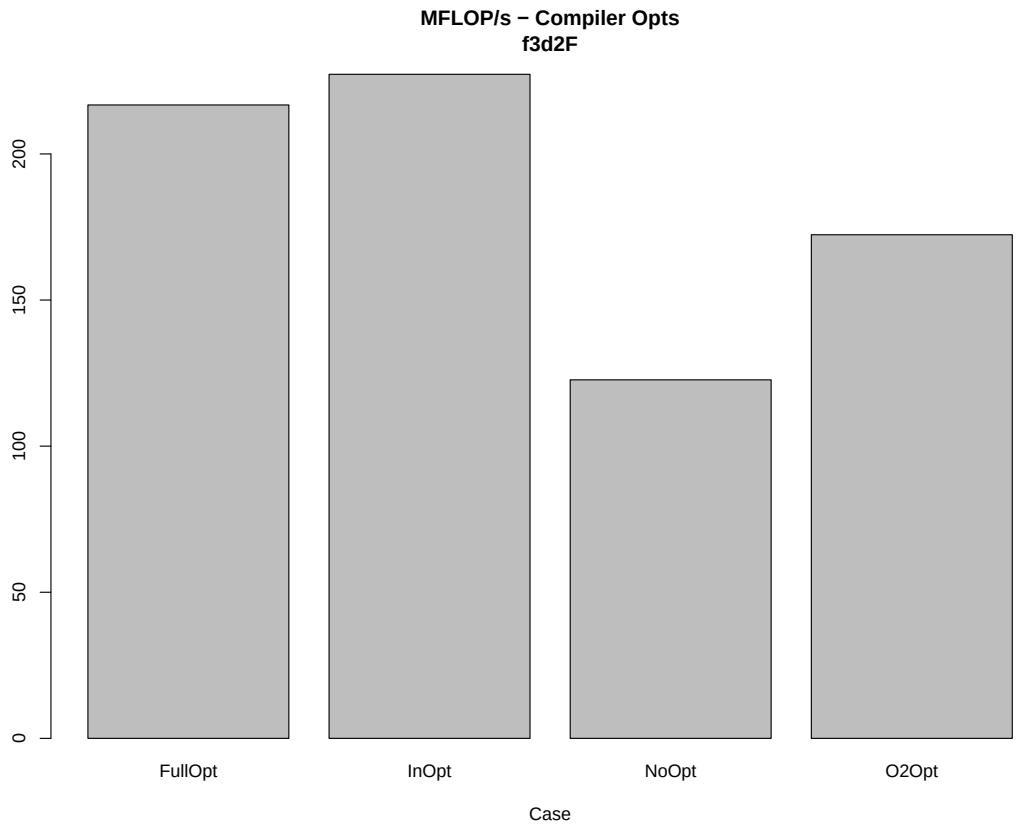



Figure 4: Performance Opt. - MFLOP/s, f3, $r = 3$, Uniform, 393216 dof

# Cache Blocking

- Following ideas of Douglas et al. (2000), the basic idea is to reuse cache levels (mainly L2) in the hardware memory hierarchy as much as possible.
- This idea can be applied in an efficient manner for routines which are repeated a fixed number of iterations over the same data, such as Gauss-Seidel used as a smoother within the Multigrid context.
- Partition the domain into $N_b$ blocks and each block into $N_c$ subblocks where $N_c = N_s + 1$, $N_s$ being the fixed number of sweeps desired.
- $N_b$ determined so that all data associated with triangles in each block will fit in L2 cache.

# Cache Blocking Partition

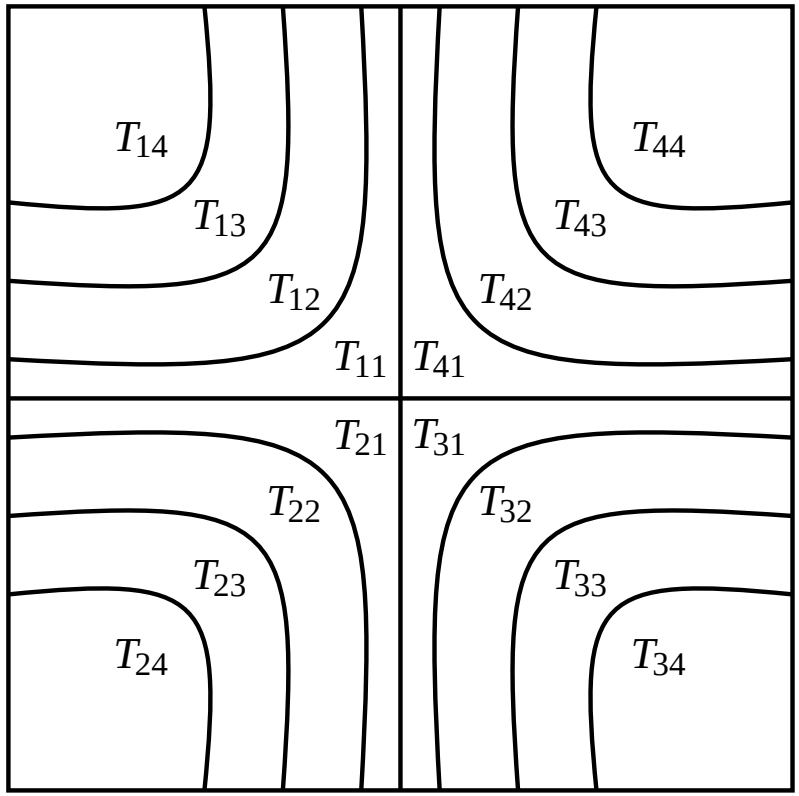
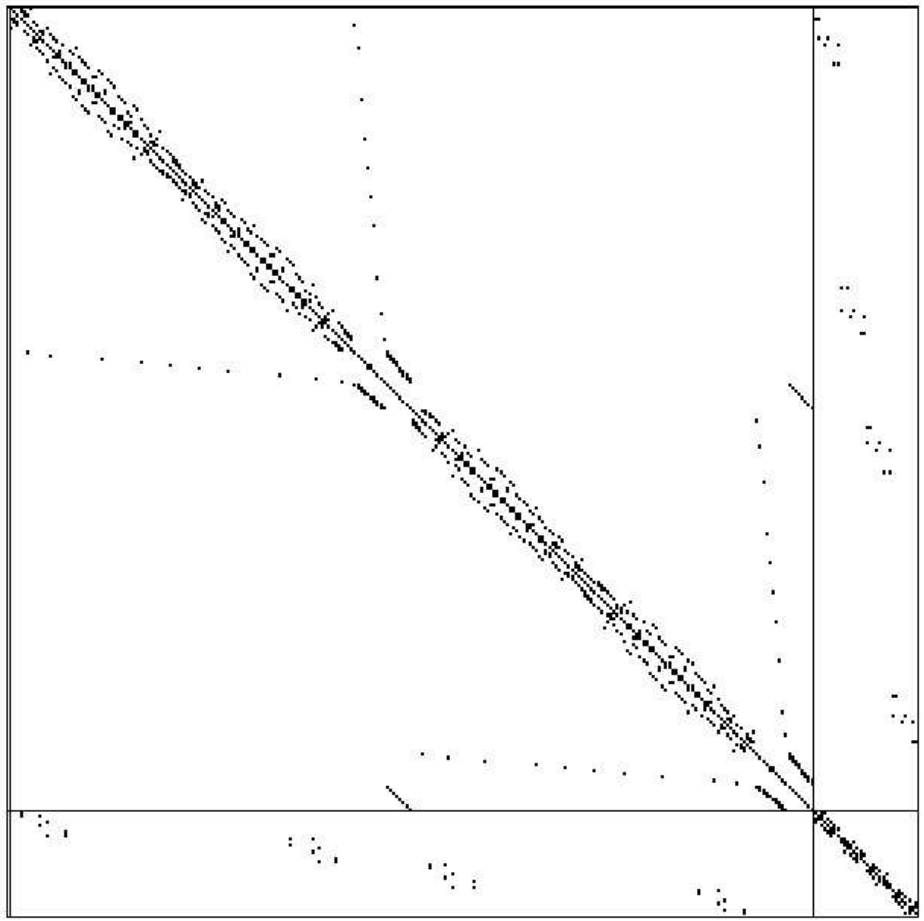



Figure 5: Block/SubBlock Partitioning for $\mathscr{T}$ with $N_b = 4, N_c = 4, N_s = 3$

# Cache Blocking Schedule

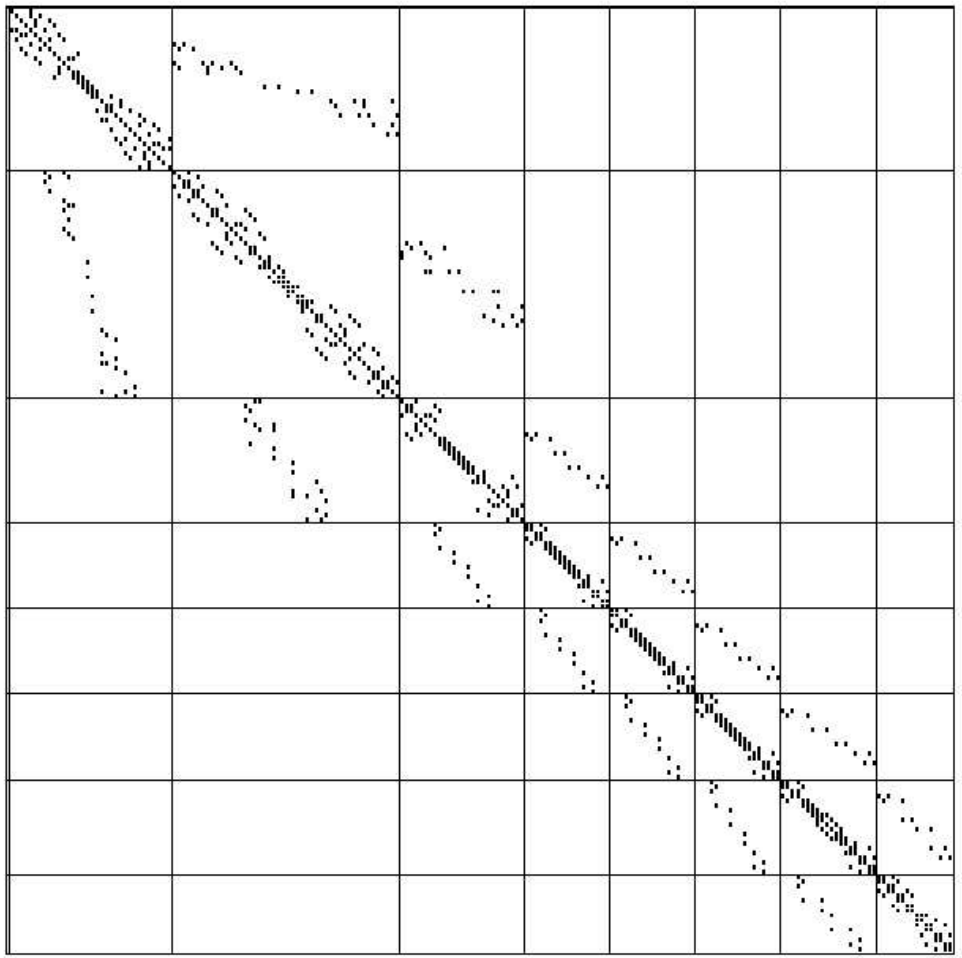
- The subblocks should be visited in the following order in order to be consistent with Gauss-Seidel sweeping through the complete domain three times:
- Primary Sweep:
  - $T_{14}, T_{13}, T_{12}, T_{11}, T_{14}, T_{13}, T_{12}, T_{14}, T_{13}$
  - $T_{24}, T_{23}, T_{22}, T_{21}, T_{24}, T_{23}, T_{22}, T_{24}, T_{23}$
  - $T_{34}, T_{33}, T_{32}, T_{31}, T_{34}, T_{33}, T_{32}, T_{34}, T_{33}$
  - $T_{44}, T_{43}, T_{42}, T_{41}, T_{44}, T_{43}, T_{42}, T_{44}, T_{43}$
- Backtracking Sweep:
  - $T_{11}, T_{12}$
  - $T_{21}, T_{22}$
  - $T_{31}, T_{32}$
  - $T_{41}, T_{42}$
  - $T_{11}$
  - $T_{21}$
  - $T_{31}$
  - $T_{41}$

# Cache Blocked Stiffness Matrix – f3



LEAF f3d1AR_A: alter = 4 L = 3 NT = 280

# Cache Blocked Stiffness Matrix – f6



LEAF f6d2AR_A: alter = 8 L = 8 NT = 198

# Cache Blocking and Data Conti-guity

- Figures 6–9 illustrates the effect of utilizing cache blocking coupled with Modified Block Sparse Row (MBSR) storage schemes.
  - NN_NN: No special performance optimizations.
  - CB_NN: Cache Blocking only.
  - NN_MB: MBSR storage.
  - CB_MB: Cache Blocking and MBSR storage.
- Note the following:
  - MBSR storage implies data contiguity of global stiffness matrix.
  - Clear benefi ts in L2 cache misses using Cache Blocking.
  - Clear benefi ts in TLB data misses using MBSR storage.
  - Clear time savings using both.

# Cache Blocking Effects

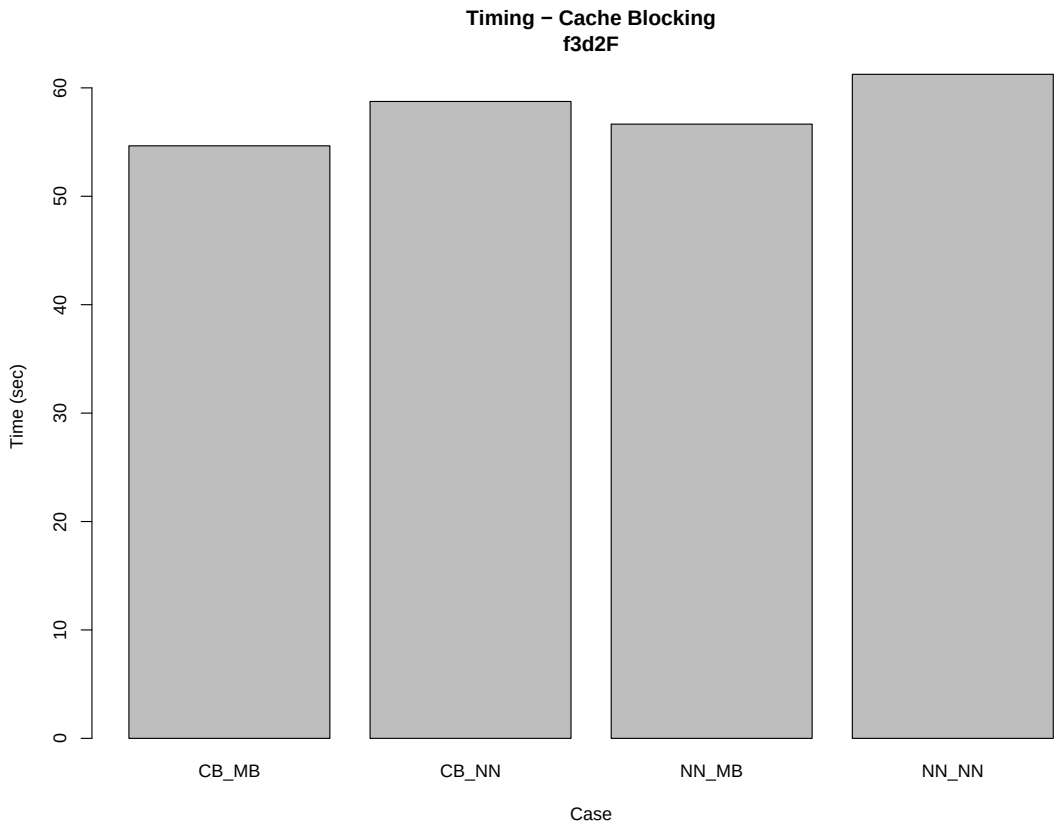



Figure 6: Cache Blocking - Time, f3, $r = 3$, Uniform

# Cache Blocking Effects, contd.

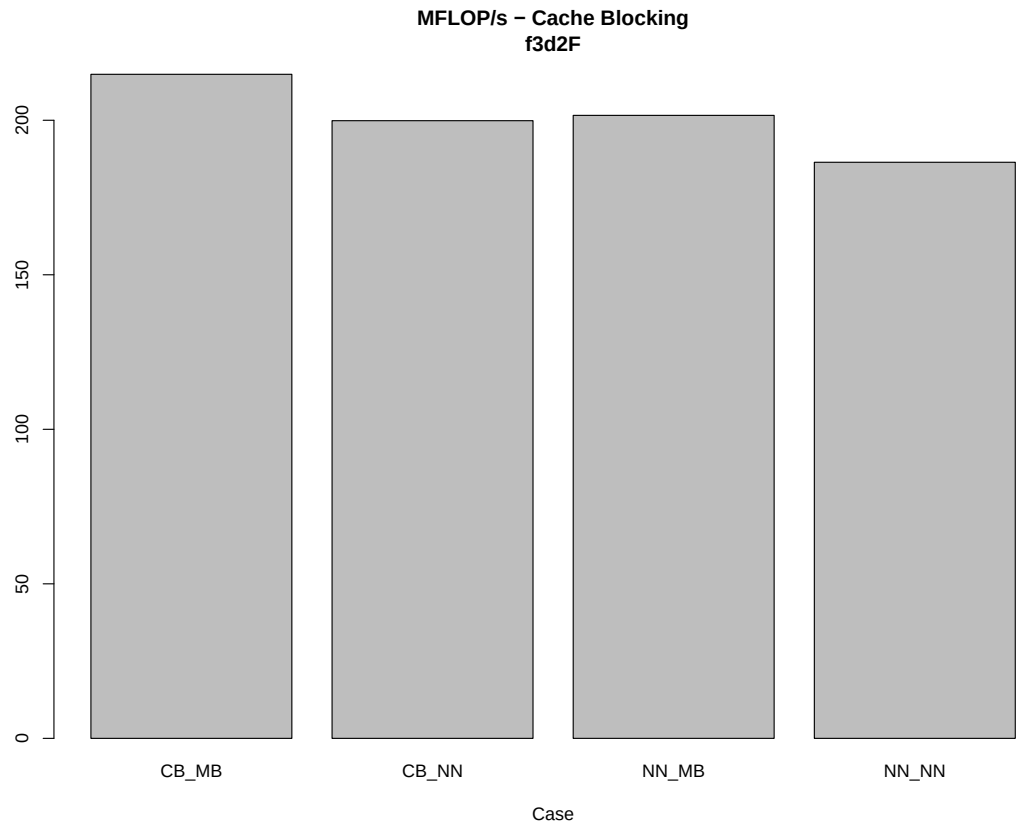



Figure 7: Cache Blocking - MFLOP/s, f3, $r = 3$, Uniform

# Cache Blocking Effects, contd.

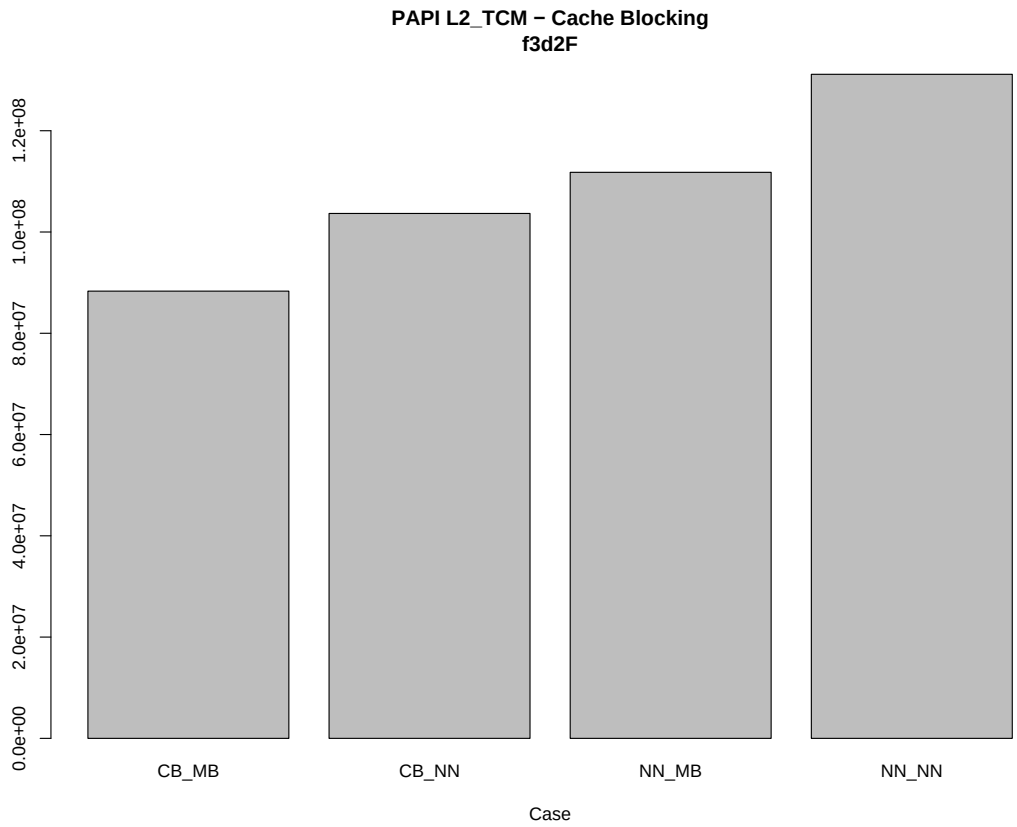



Figure 8: Cache Blocking - L2_TCM, f3, $r = 3$, Uniform

# Cache Blocking Effects, contd.

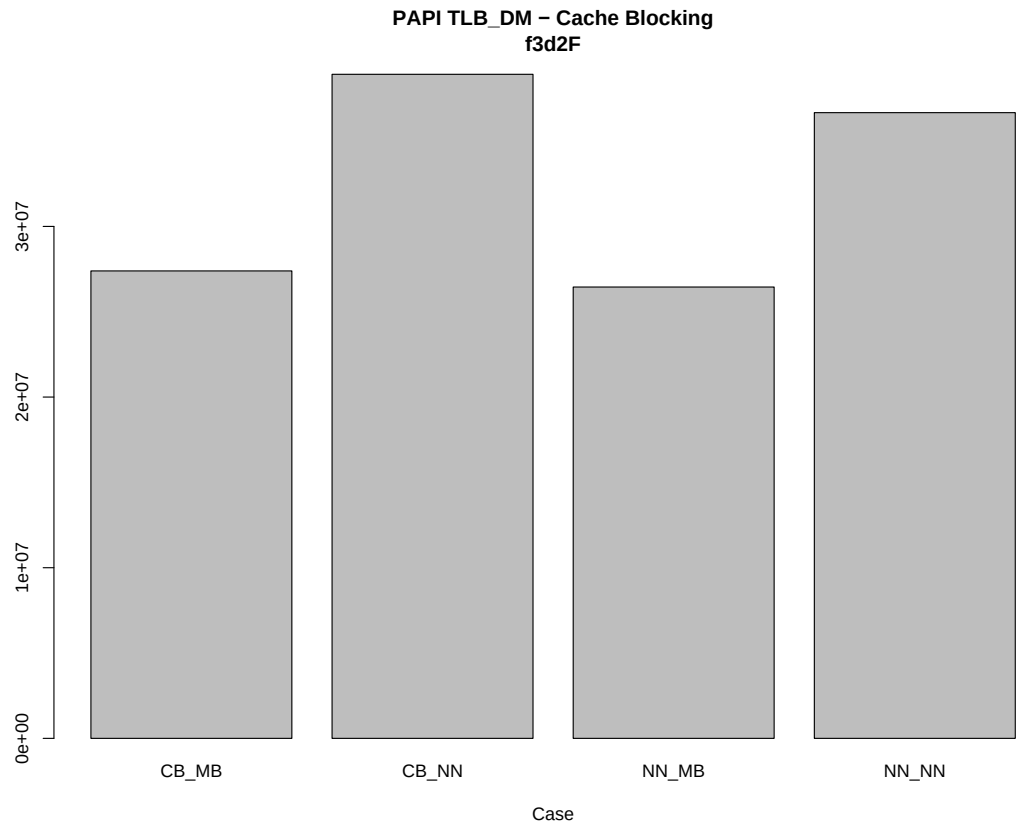
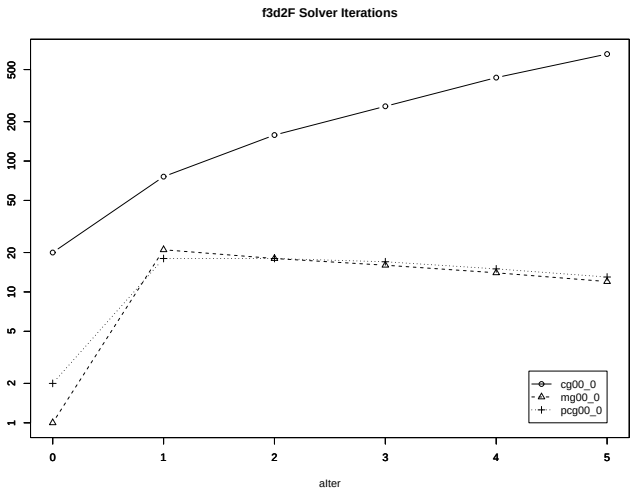



Figure 9: Cache Blocking - TLB_DM, f3, $r = 3$, Uniform

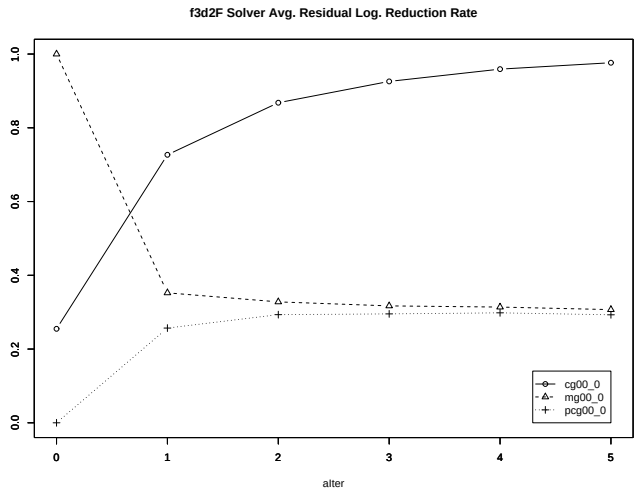
# Linear Solvers

- Three different iterative solvers can be used to solve the resulting linear systems.
- Conjugate Gradient (CG), Multigrid (MG), and Preconditioned Conjugate Gradient (PCG/MG) using MG as a preconditioner.
- MG (and PCG/MG) require a fi xed number of *smoothing* sweeps to dampen out low frequency error components during the multigrid procedure. Cache blocked Gauss-Seidel works well in this regard.

# Linear Solver Comparisons



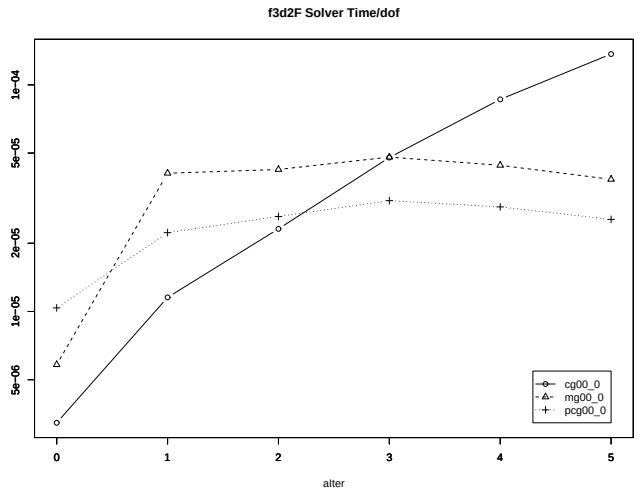

(a) Solver Iterations



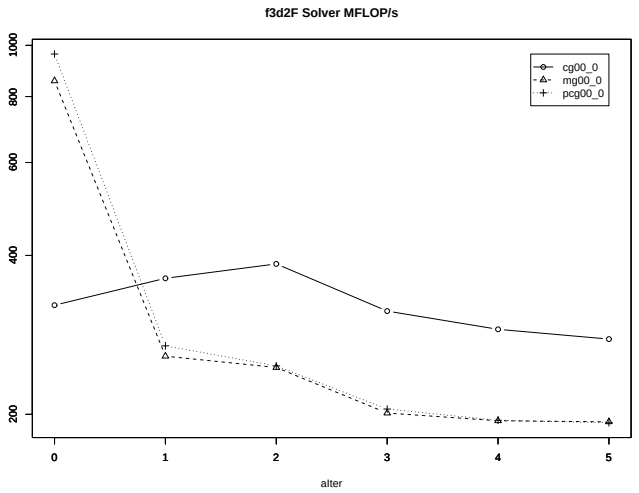

(b) Solver Avg. Log. Residual Reduction Rate

Figure 10: Solver Comparison (Effi ciency): f3, $r = 3$, Uniform

# Lin. Solver Comp., contd



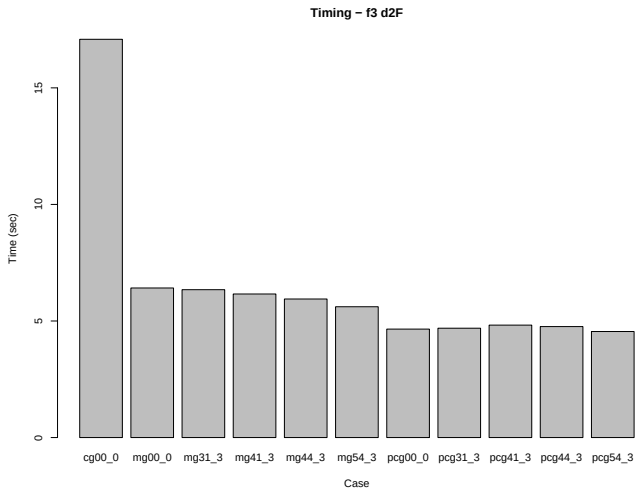

(a) Solver Time/dof



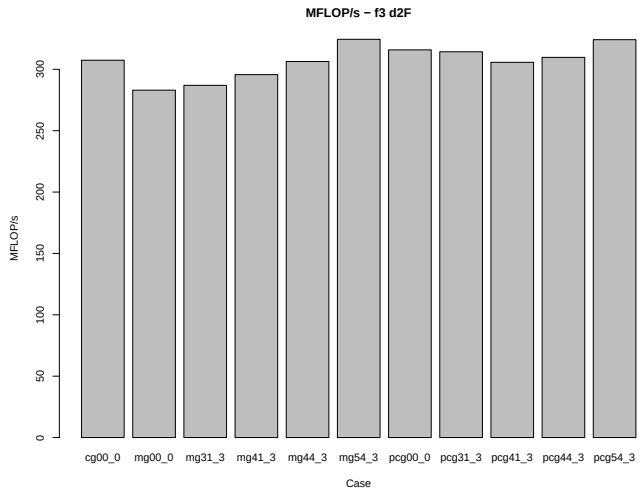

(b) Solver MFLOP/s

Figure 11: Solver Comparison (Timing): f3, $r = 3$, Uniform

# Lin. Solver Comp., contd




(a) Solver Time




(b) Solver MFLOP/s

Figure 12: Solver Optimization Comparison : f3, $r = 3$, Uniform

# Ell2 Results

What follows are selected charts and graphs illustrating different aspects of Ell2 code performance.

- A priori error reduction in the energy norm under uniform refi nement.
- Adaptive error reduction in the energy norm.
- Effectivity indices for the residual estimator described above.
- Adaptive meshes.

# Ell2 Error Energy Norm

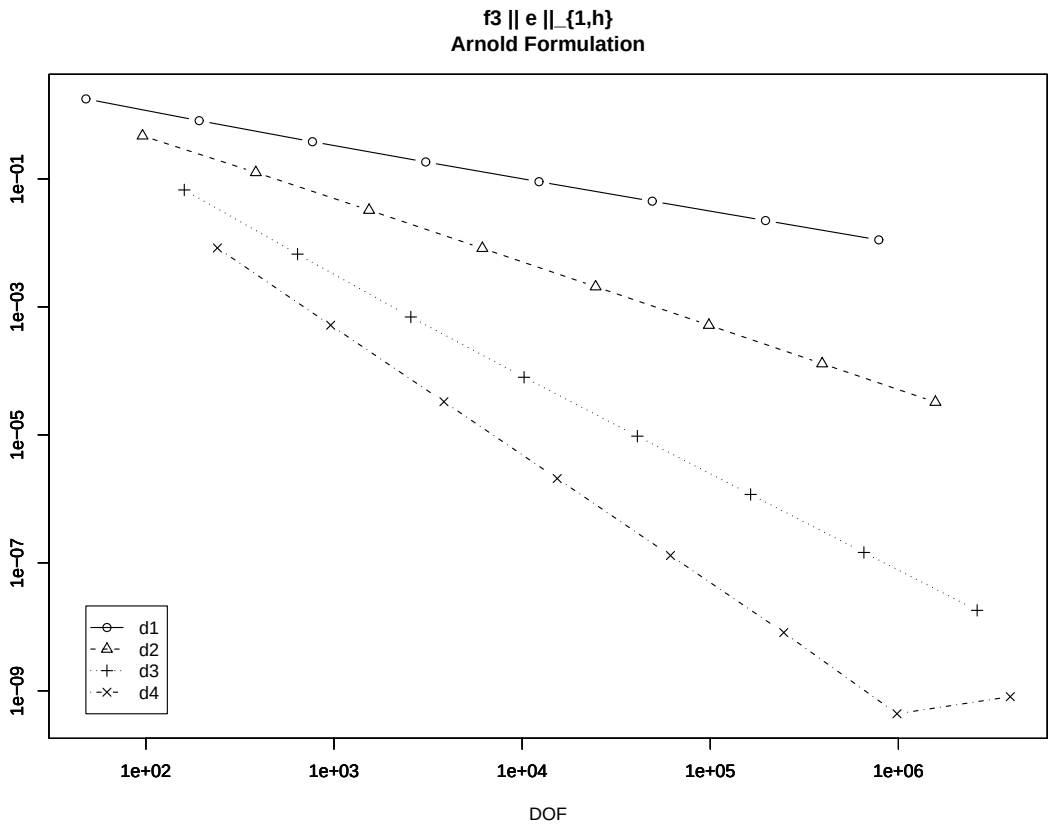



Figure 13: Uniform Energy Norm: f3, $r = 3$, Arnold

# Ell2 Err. Energy, contd.

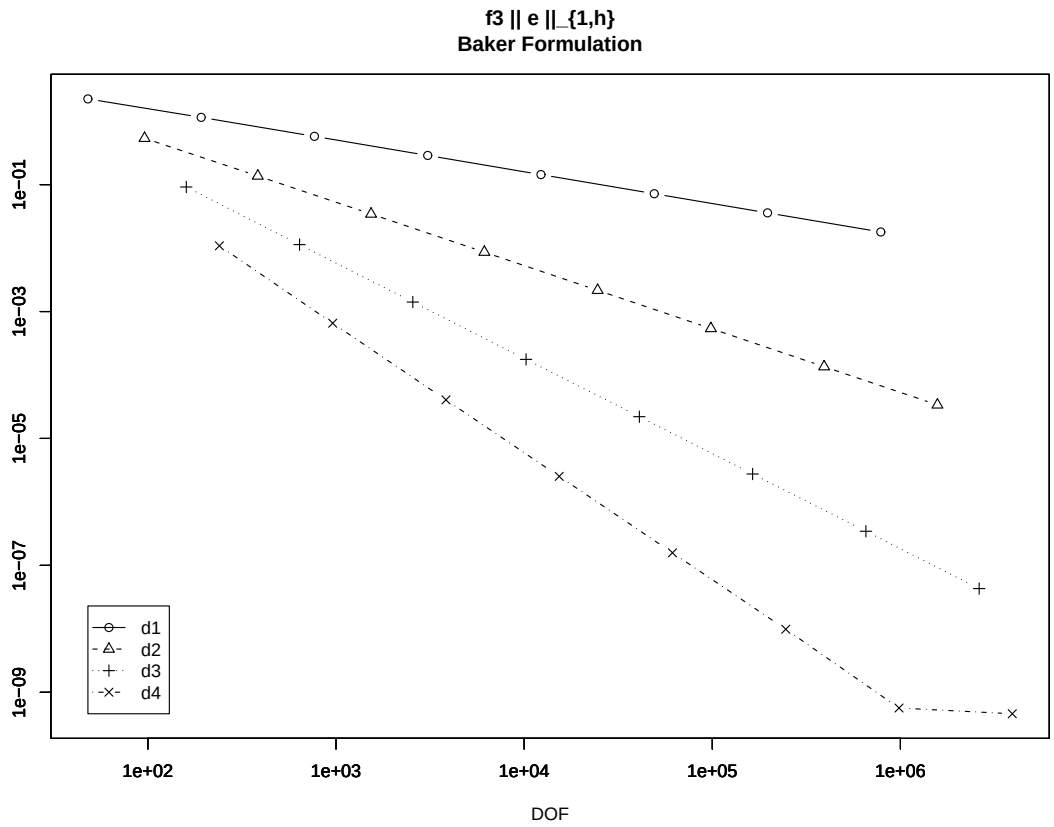



Figure 14: Uniform Energy Norm: f3, $r = 3$, Baker

# Ell2 Err. Energy, contd.

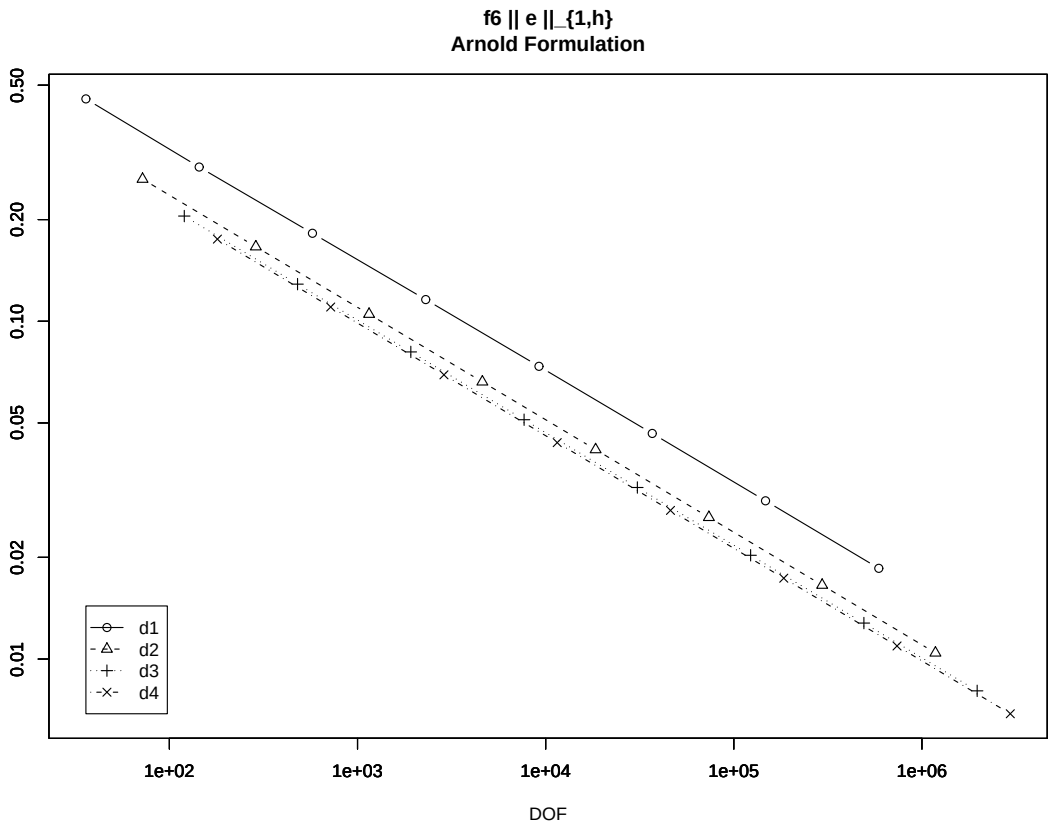



Figure 15: Uniform Energy Norm: f6, $r = 3$, Arnold

# Ell2 Err. Energy, contd.

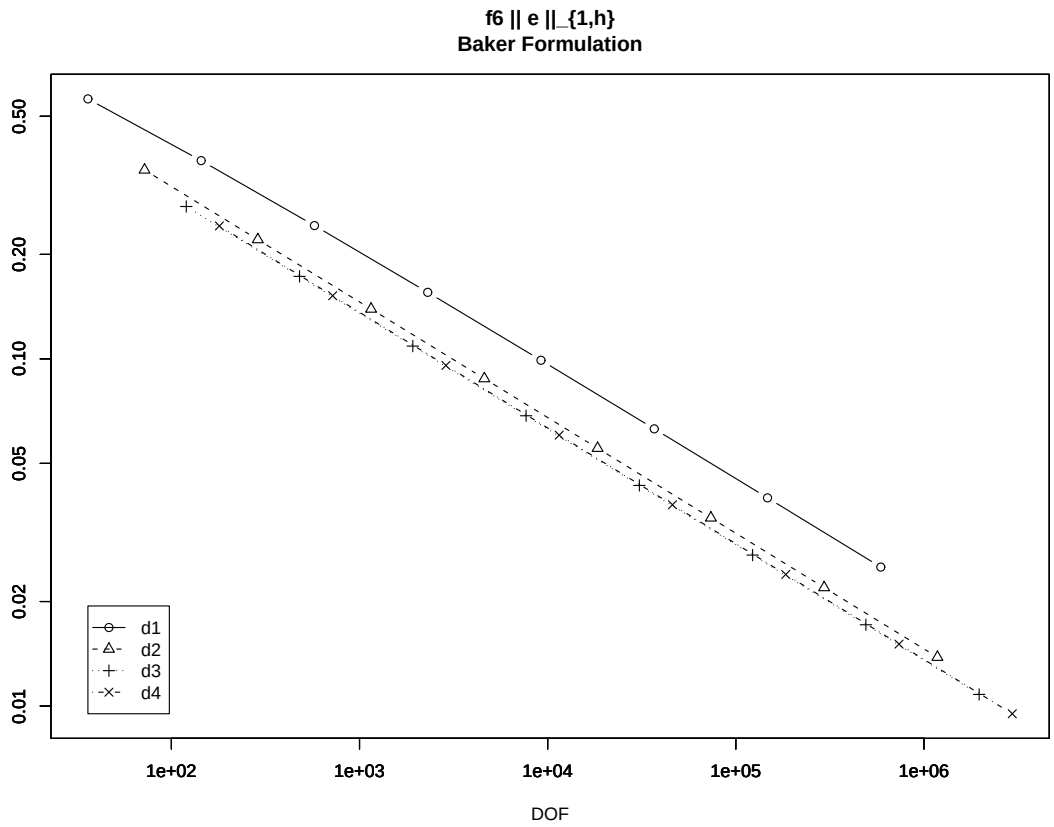



Figure 16: Uniform Energy Norm: f6, $r = 3$, Baker

# Ell2 Adaptive Error Energy Norm

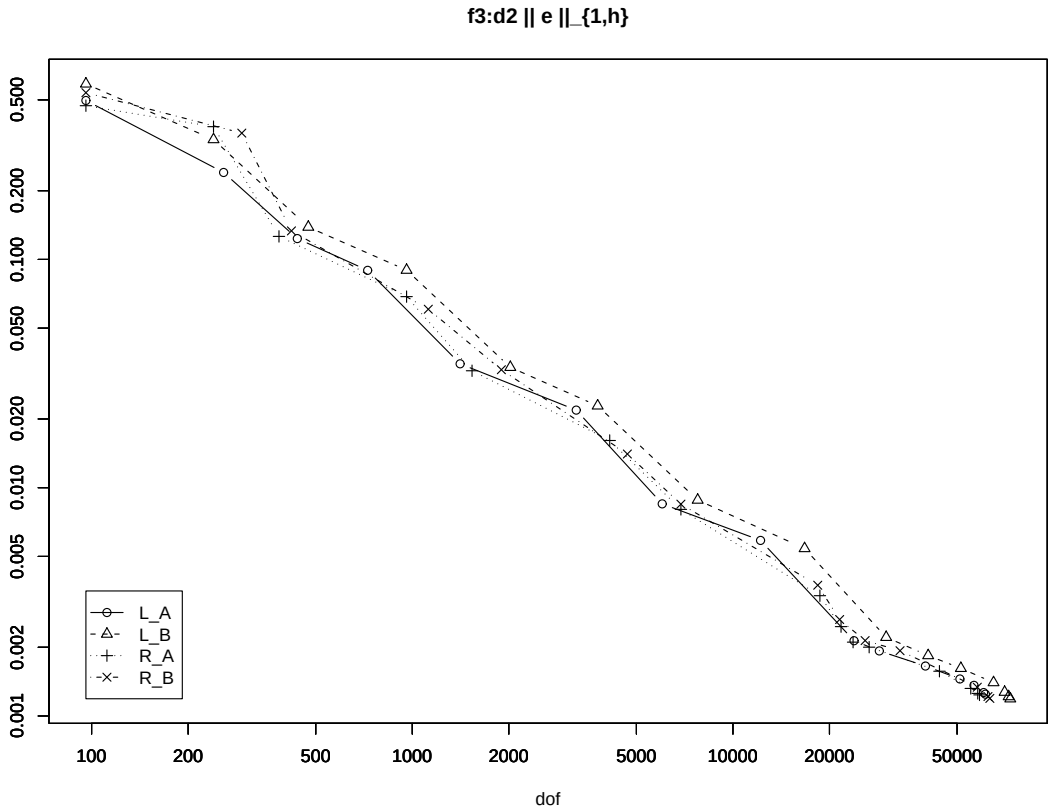



Figure 17: Adaptive Error Energy Norm: f3, $r = 3$

# Ell2 Adapt. Err. Energy, contd.

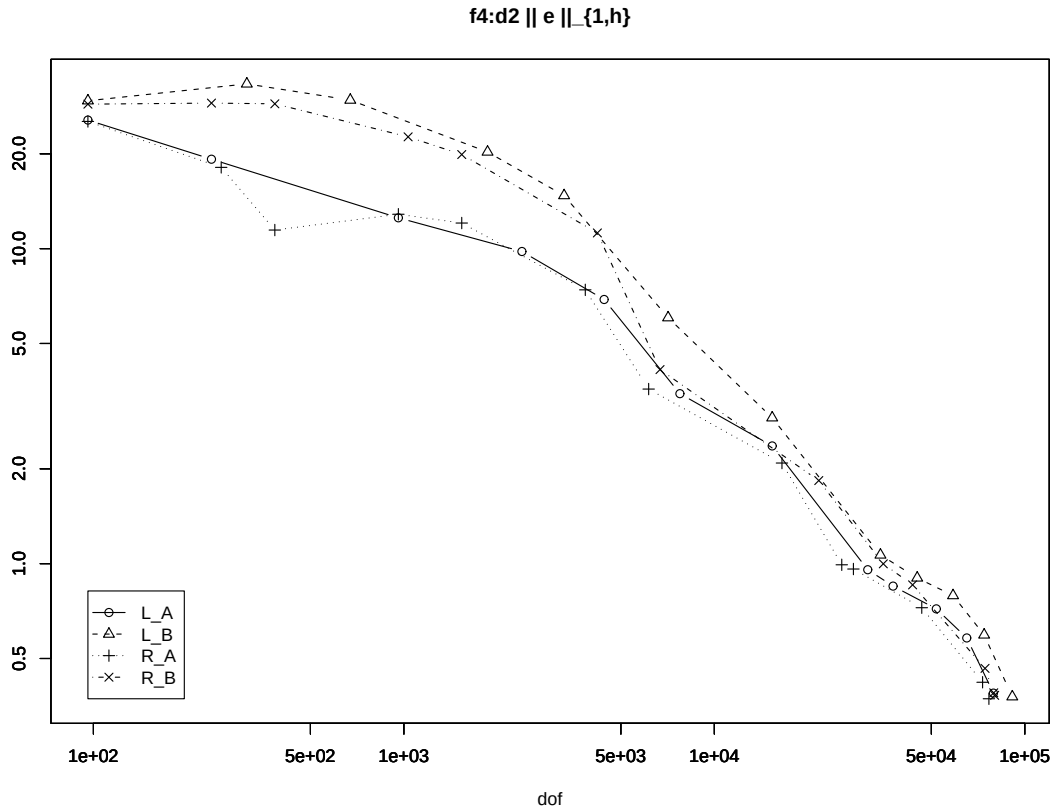



Figure 18: Adaptive Error Energy Norm: f4, $r = 3$

# Ell2 Adapt. Err. Energy, contd.

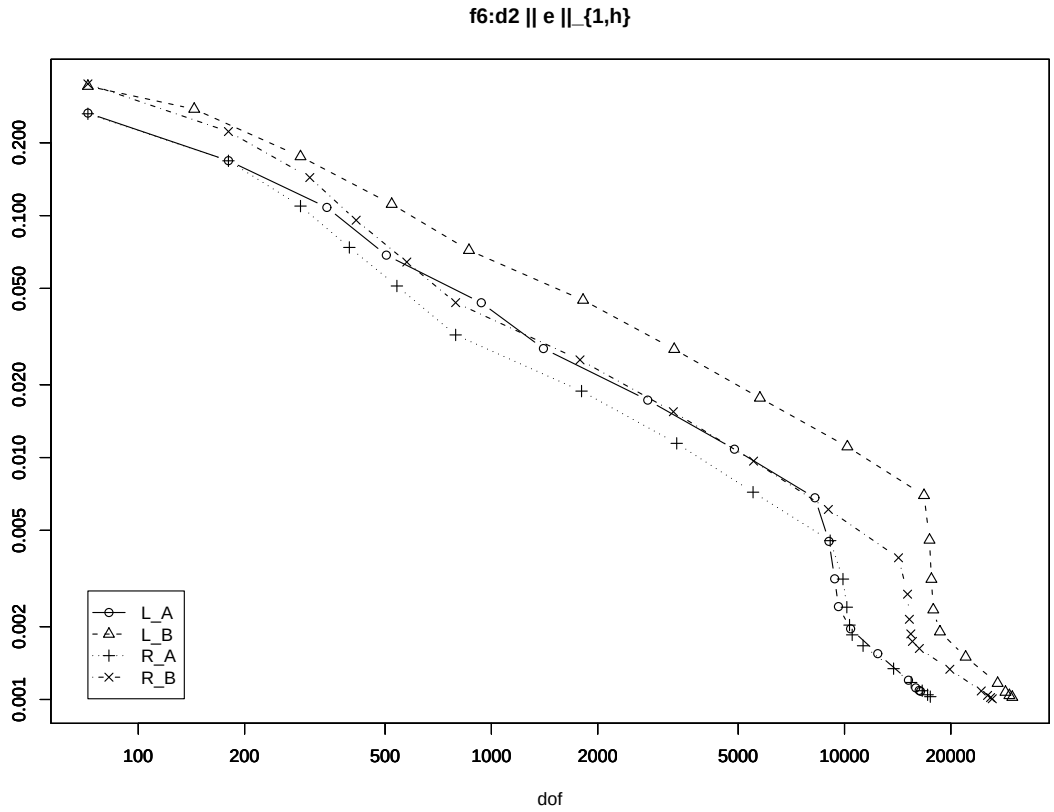



Figure 19: Adaptive Error Energy Norm: f6, $r = 3$

# Effectivity Indices

- Effectivity indices give insight as to how well the estimator tracks the actual error.
- The following effectivity indice graphs chart the effectivity index defi ned for the residual estimator

$$\eta_{\mathscr{T}_h} = \left( \sum_{K \in \mathscr{T}_h} \eta_K^2 \right)^{1/2} \quad \text{as}$$

$$\eta = \frac{\eta_{\mathscr{T}_h}}{\|e\|_{1,h}}.$$

# Ell2 Eff. Ind., contd.

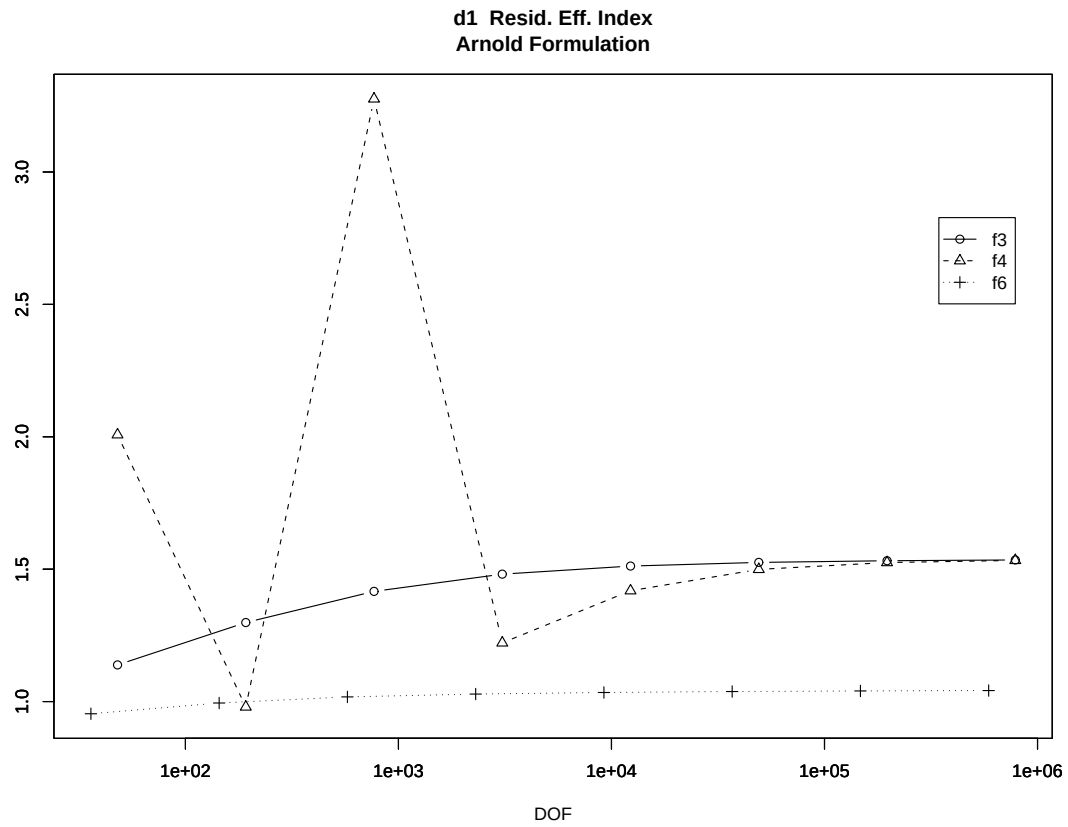



Figure 20: Effectivity Indices, $r = 2$, Arnold

# Ell2 Eff. Ind., contd.

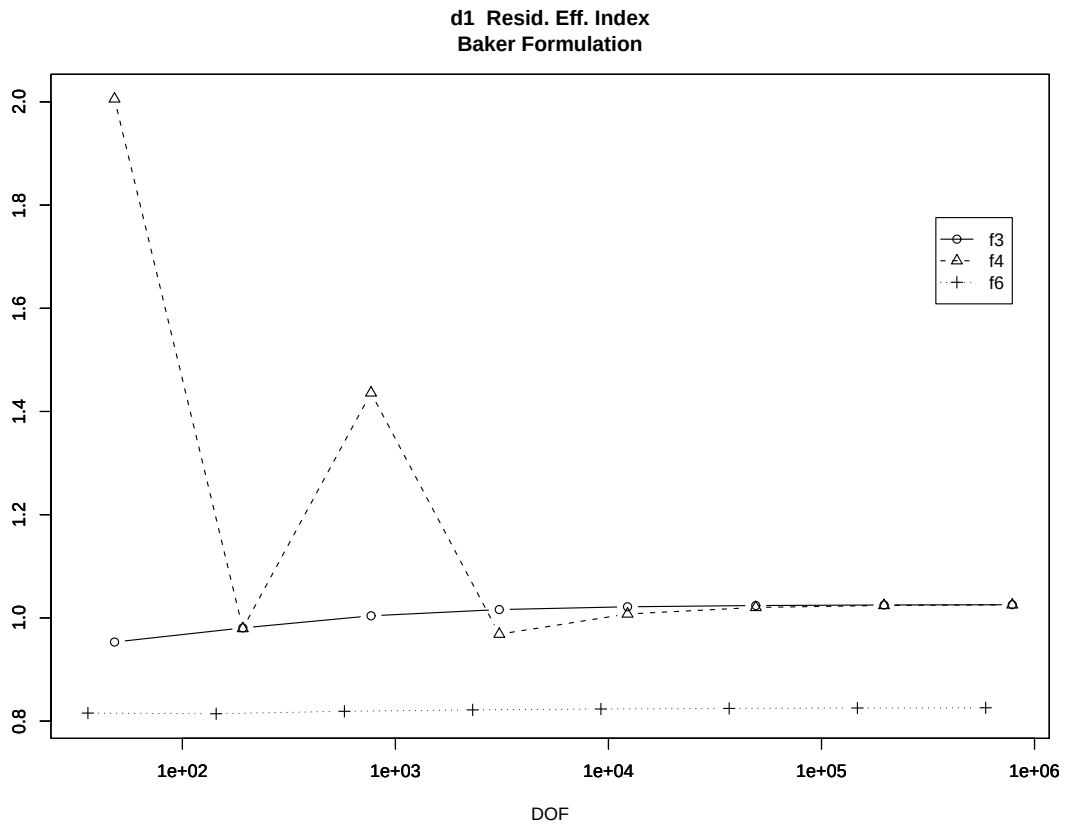



Figure 21: Effectivity Indices, $r = 2$, Baker

# Ell2 Eff. Ind., contd.

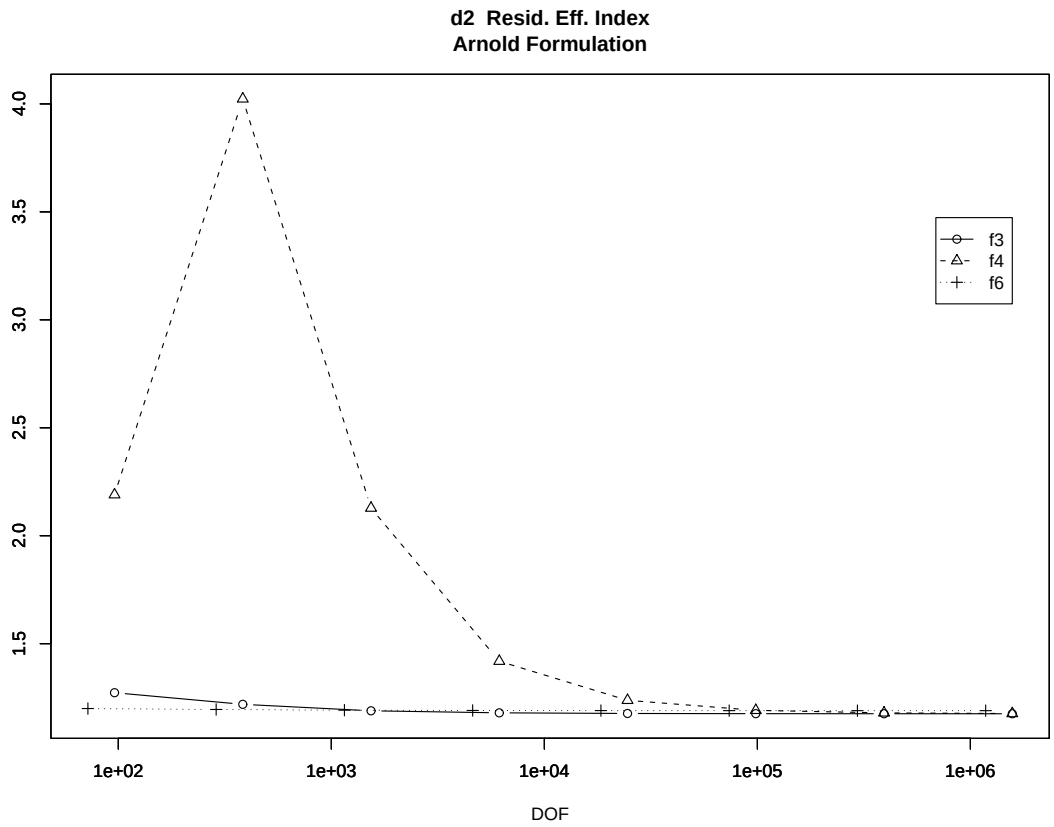



Figure 22: Effectivity Indices, $r = 3$, Arnold

# Ell2 Eff. Ind., contd.

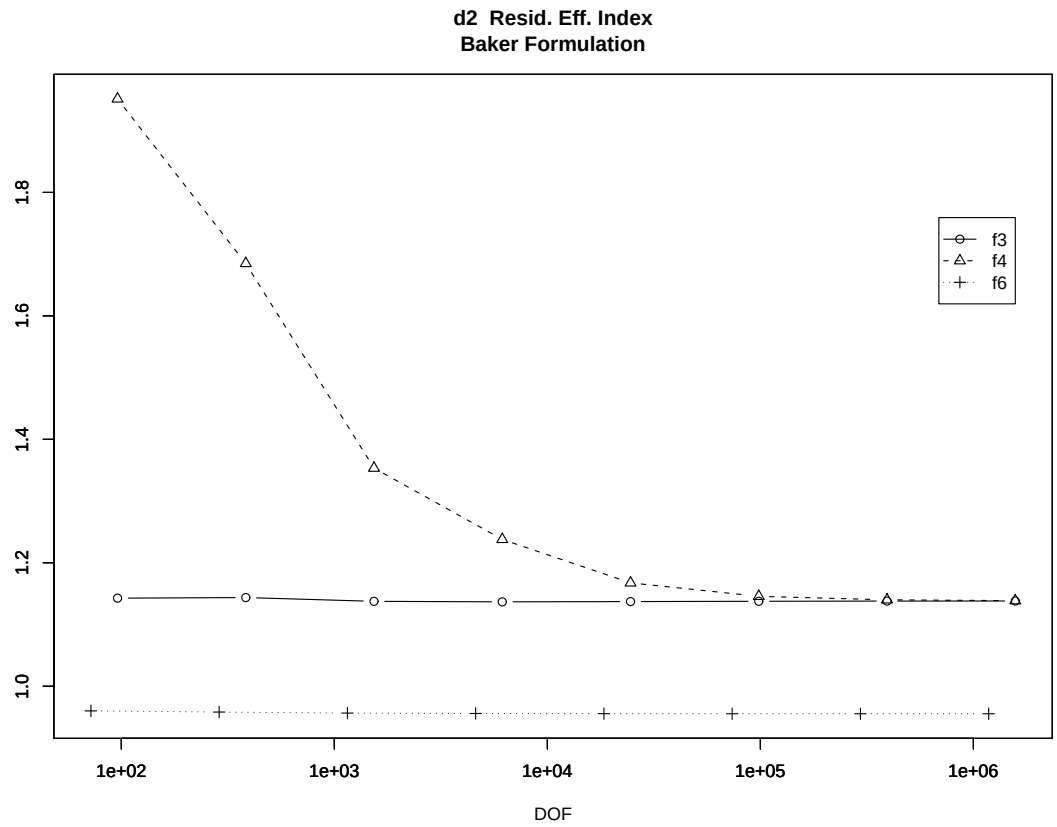



Figure 23: Effectivity Indices, $r = 3$, Baker

# Ell2 Eff. Ind., contd.

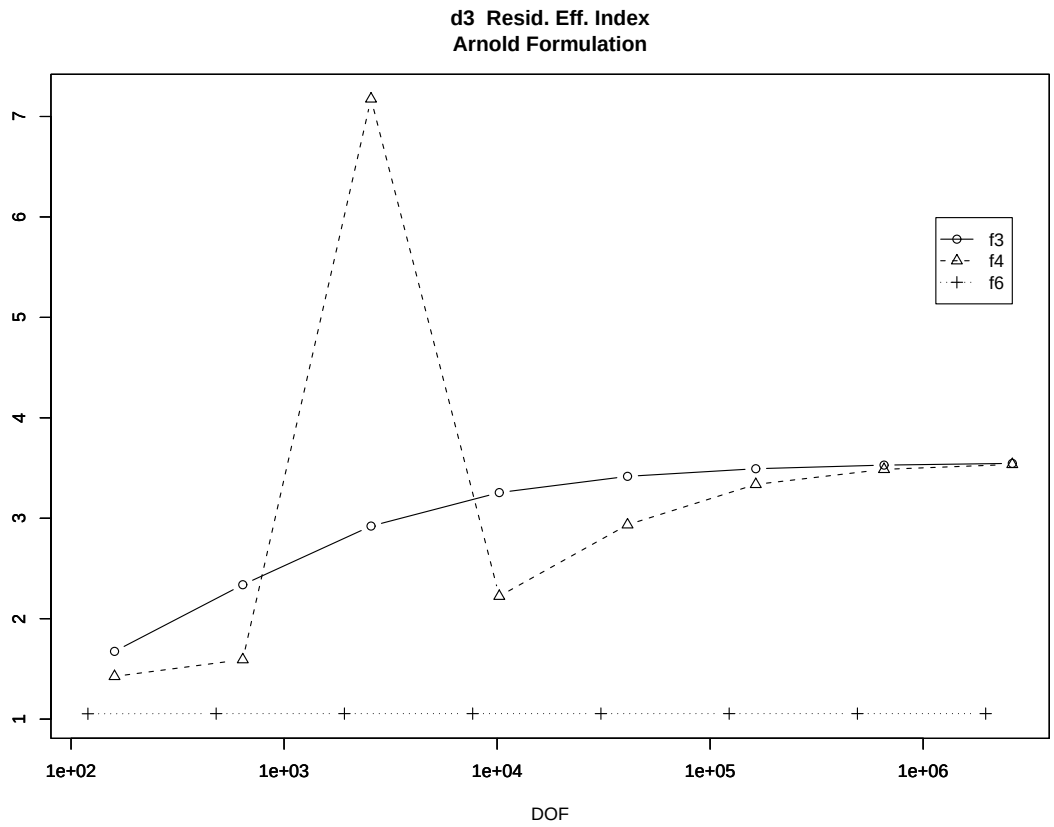



Figure 24: Effectivity Indices, $r = 4$, Arnold

# Ell2 Eff. Ind., contd.

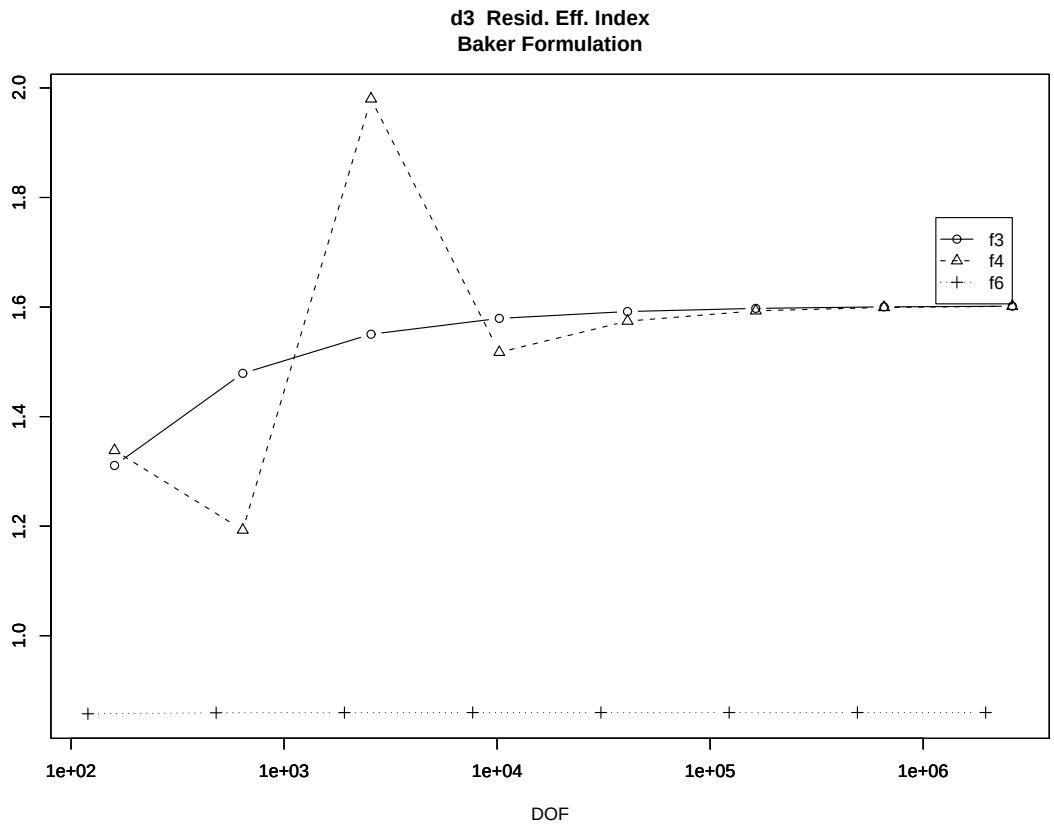



Figure 25: Effectivity Indices, $r = 4$, Baker

## Estimators and Adaptive Meshes

- f3d1a est – UNIX
- f3d1a est – Windoze
- f3d2a est – UNIX
- f3d2a est – Windoze
- f4d1a est – UNIX
- f4d1a est – Windoze
- f6d2a est – UNIX
- f6d2a est – Windoze

# PLTMG Comparison

- A comparison with PLTMG (Bank, 1998) indicates better performance of DG-Ell2 compared to PLTMG.
- PLTMG: f4, 31561 linear triangular elements, 16000 dof, $\|\nabla e\| \approx 9.99$, $\|e\| \approx 5.24e-2$, 4.1 sec.
- Ell2: f4, 5431 linear triangular elements, 16023 dof, $\|\nabla e\| \approx 3.39$, $\|e\| \approx 1.8e-2$, 4.2 sec.

# PLTMG Comp, contd.

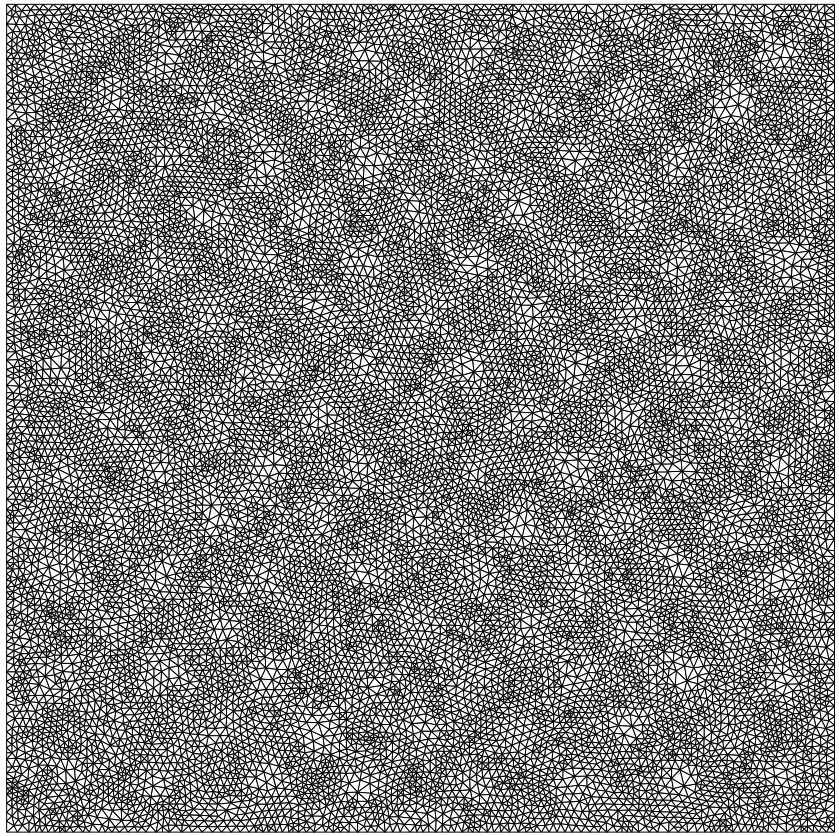


Figure 26: PLTMG: f4, $r = 2$

# PLTMG Comp, contd.

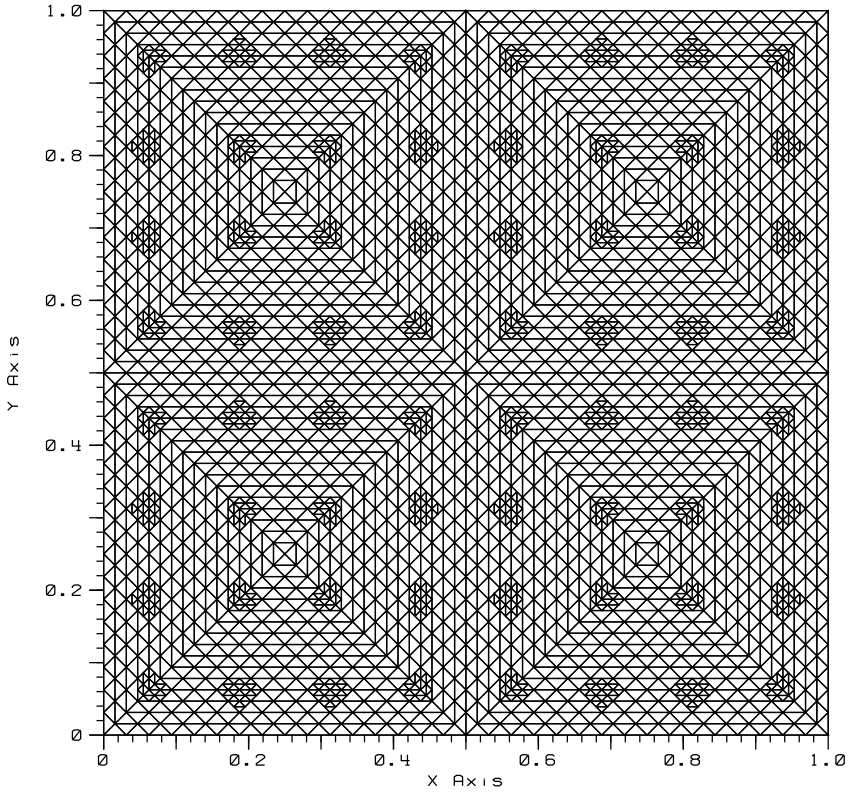



Figure 27: Ell2: f4, $r = 2$

# Ell4 – Results

- The biharmonic problem is more difficult to solve than second order elliptic because the stiffness matrix condition number grows as $O(h^{-4})$.
- For Ell4 test problems, typical to require between 50–150 PCG solver iterations to obtain accuracy of $10^{-13}$, compared with 10–20 PCG solver iterations to reach same accuracy for Ell2 test problems.
- Have implemented a variable V-cycle version of multilevel solver, increasing the number of smoother iterations the coarser the mesh, similar to that employed by Gopalakrishnan and Kanschat (2003).

# Ell4 – Test Problem f2

Test Problem - f2 Domain $\Omega$: Figure 28

$$\begin{cases} \Delta^2 u = 288x^2y^2 - 48y + 8 + 72x^2 + 24y^4 - 288x^2y \\ \qquad\qquad +72y^2 - 288xy^2 + 288xy - 48y^3 - 48x + 24x^4 - 48x^3 & \text{in } \Omega \\ u = \partial_n u = 0 & \text{on } \Gamma \end{cases}$$

Exact solution: $u = x^2y^2\,(1-x)^2\,(1-y)^2$.

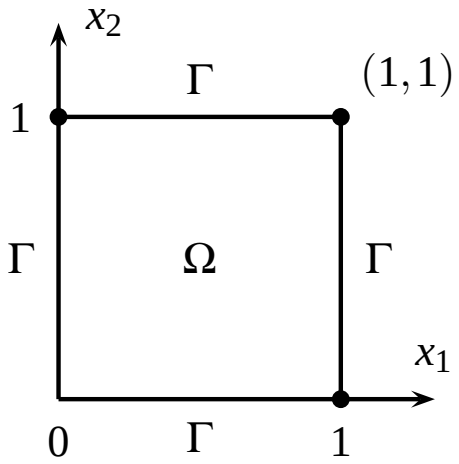



Figure 28: Square Domain

# Ell4 – f2 Exact Solution

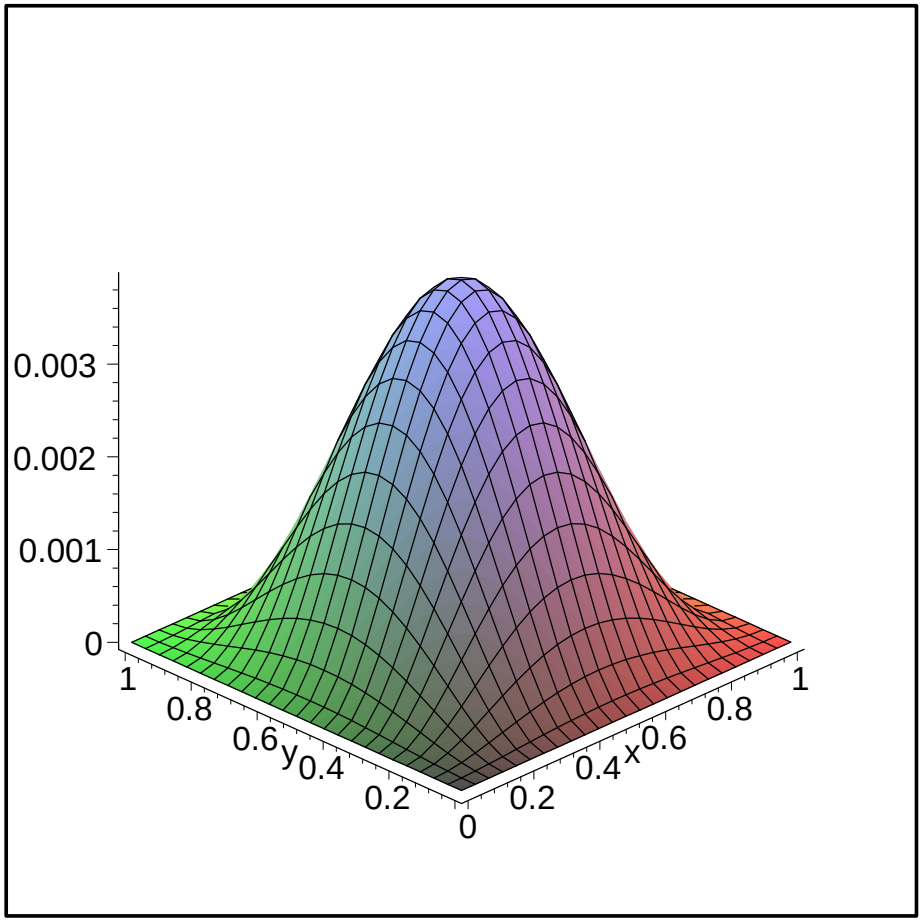
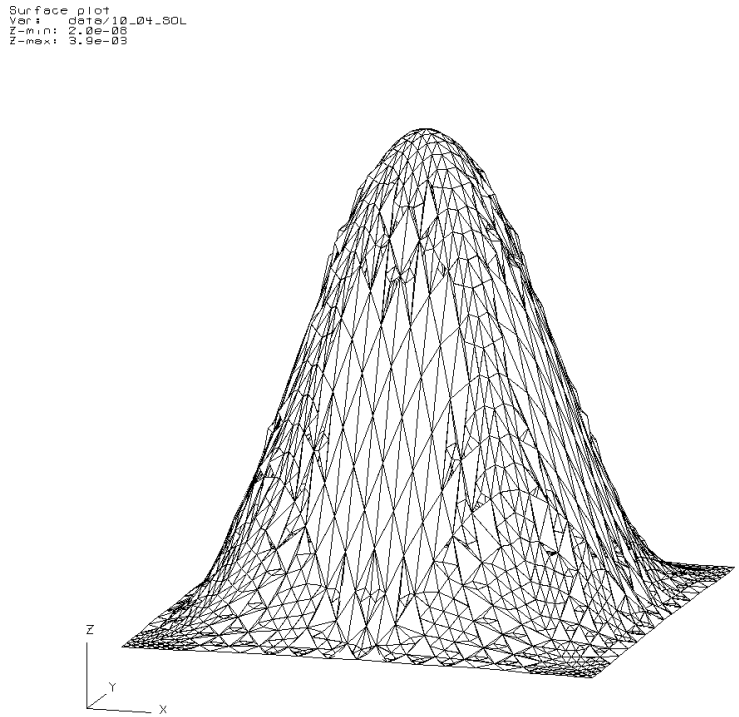
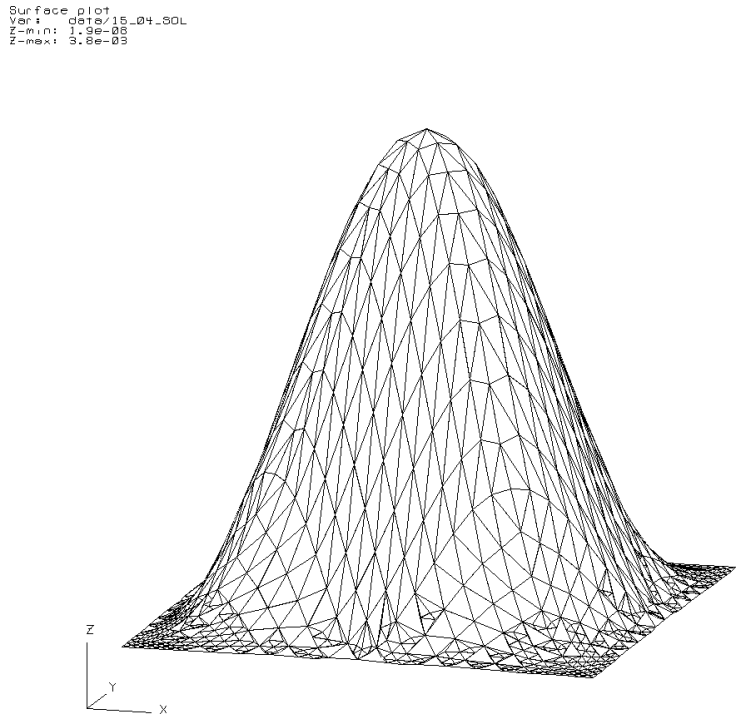



Figure 29: f2 Exact Solution

# Ell4 – Computed Solution, f2




(a) Ell4: f2, $r = 4$




(b) Ell4: f2, $r = 5$

Figure 30: Ell4: f2

# Ell4 – Test Problem f4

## Test Problem - f4

Domain $\Omega$: Figure 28

$$\begin{cases} \Delta^2 u = -16\cos(2\pi x)\pi^4 + 64\cos(2\pi x)\pi^4\cos(2\pi y) - 16\cos(2\pi y)\pi^4 & \text{in } \Omega \\ u = \partial_n u = 0 & \text{on } \Gamma \end{cases}$$

Exact solution: $u = (1-\cos(2\pi x))\,(1-\cos(2\pi y))$.

# Ell4 – f4 Exact Solution

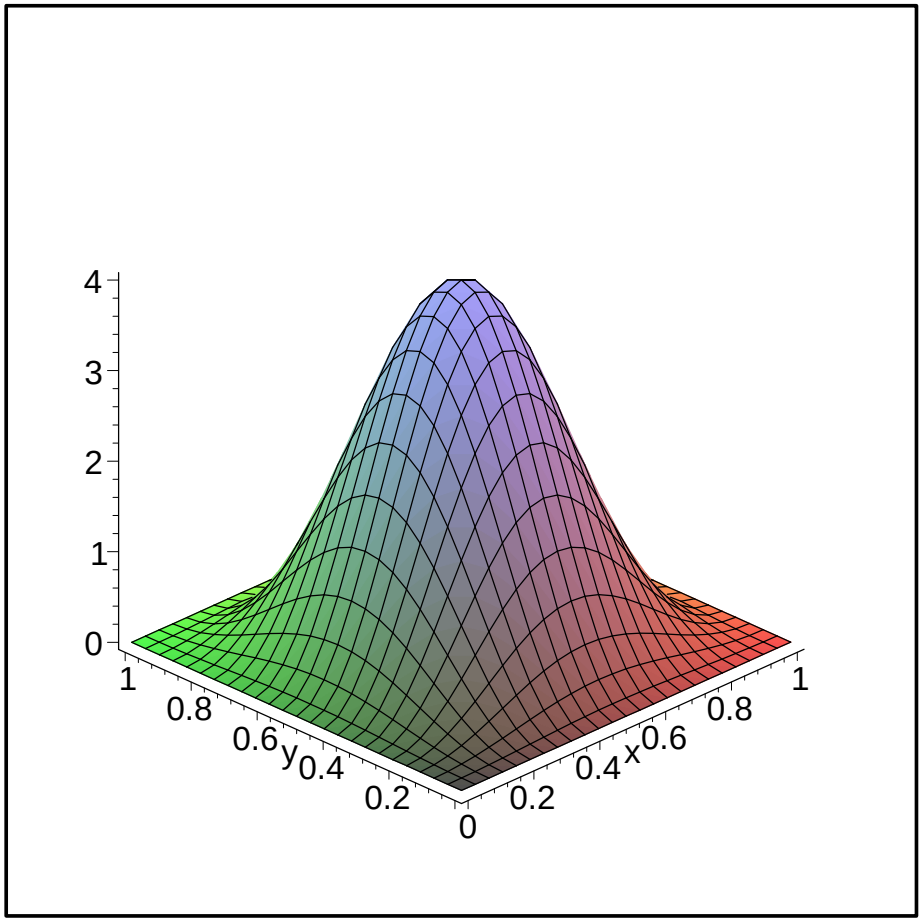
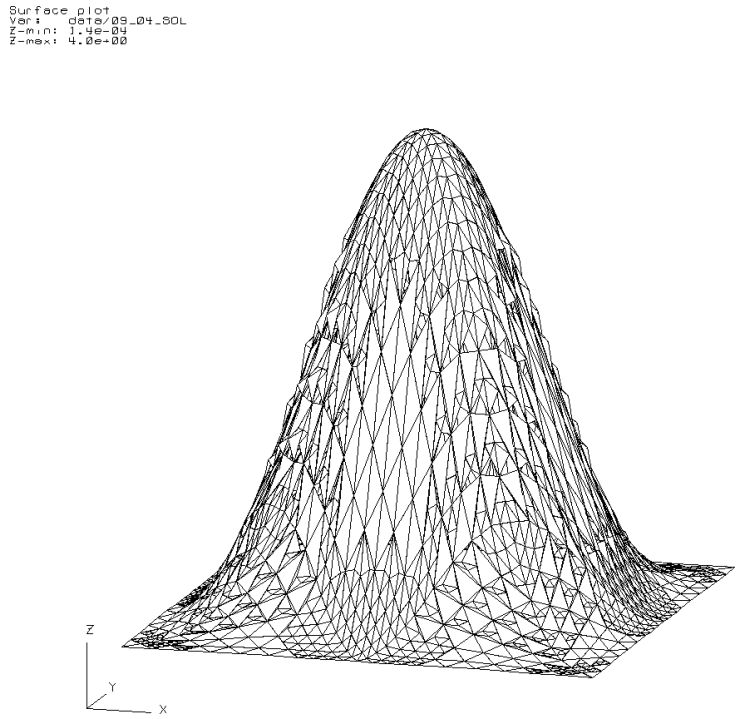
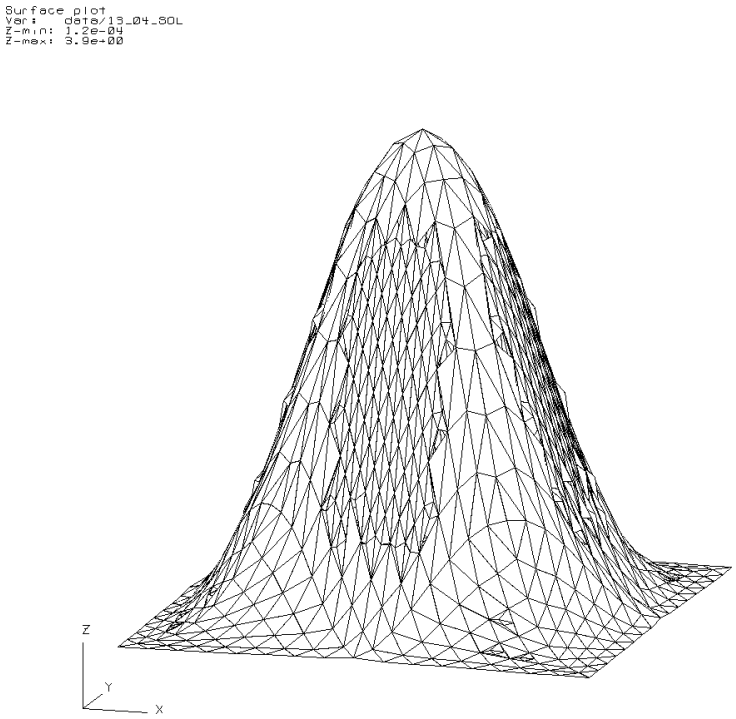



Figure 31: f4 Exact Solution

# Ell4 – Computed Solution, f4




(a) Ell4: f4, $r = 4$




(b) Ell4: f4, $r = 5$

Figure 32: Ell4: f4

# Future Directions

- Perform more extensive comparisons with other state of the art FEM computer codes such as KASKADE, Alberta, and deal.II.
- Improving data object and list level management in adaptive environments.
- Integration of cache blocking concepts to the full mesh hierarchy with Multigrid.
- Further optimization for L1 cache.
- Tuning of variable $\theta$ algorithms for marking.
- Include coarsening for elliptic problems.
- Move to time dependent problems, including parabolic and Cahn-Hilliard.

# Future Directions

- Incorporate nonlinear solvers to handle nonlinear PDEs.
- Develop new sharp a posteriori estimates.
- Develop a “drastic cutting” strategy to reduce number of adaptive iterations and quickly “zoom” in to the solution.
- Implement $h$–$p$ a posteriori error estimators, i.e., make a determination to refine/coarsen in space or finite element polynomial degree, or both.
- Identify and implement new optimization techniques for Multigrid, including preconditioners.
- Investigate the use of space-filling curves to obtain optimal ordering for the various algorithms.
- Extend 2D results to 3D results.

# References


Arnold, D. (1982). An interior penalty finite element method with discontinuous elements. *SIAM J. Num. Anal.*, 19:742–760.

Babuşka, I. and Strouboulis, T. (2001). *Finite Element Method and its Reliability*. Numerical Mathematics and Computation. Oxford University Press, New York.

Baker, G. (1977). Finite element methods for elliptic equations using nonconforming elements. *Math. Comp.*, 31:45–59.

Bank, R. E. (1998). *PLTMG: A software package for solving elliptic partial differential equations, Users' Guide 8.0*. SIAM, Philadelphia.

Dörfler, W. (1996). A convergent adaptive algorithm for poisson's equation. *SIAM J. Numer. Anal.*, 33:1106–1124.

Douglas, C. C., Hu, J., Kowarschik, M., Rüde, U., and Weiss, C. (2000). Cache optimization for structured and unstructured grid multigrid. *Elect. Trans. Numer. Anal.*, 10:21–40.

Gopalakrishnan, J. and Kanschat, G. (2003). A multilevel discontinouus Galerkin method. *Numer. Math.*, 95:527–550.

Karakashian, O. and Pascal, F. (2003). A posteriori error estimates for a discontinuous Galerkin approximation of second-order elliptic equations. *SIAM J. Num. Anal.*, 41:2374–2399.

Karakashian, O. and Pascal, F. (2004). Adaptive discontinuous Galerkin approximations of second-order elliptic equations. In et al., P. N., editor, *In Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2004*, Jyväskylä, Finland.

Karakashian, O. and Pascal, F. (2006). Adaptive discontinuous Galerkin approximations of second-order elliptic problems. *SIAM J. Numer. Anal. (to appear)*.

Verfürth, R. (1995). *A review of A posteriori error Estimation and Adaptive Mesh Refinement Techniques*. Wiley-Teubner, New York.