Comparison of *hp*-Adaptive Finite Element Strategies

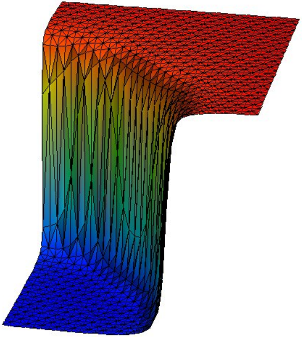
William F. Mitchell
Marjorie A. McClain

Applied and Computational Mathematics Division
National Institute of Standards and Technology
Gaithersburg, MD

March 13, 2012
ACMD Seminar Series

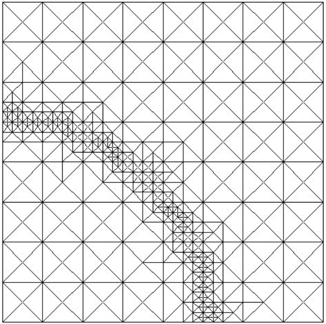
hp-Adaptive finite elements

- The finite element method approximates the solution, u, of a partial differential equation by a continuous piecewise polynomial function, u_{hp}
- u_{hp} is a polynomial over each element (triangle) of a grid
- the polynomial degree may be different over different elements



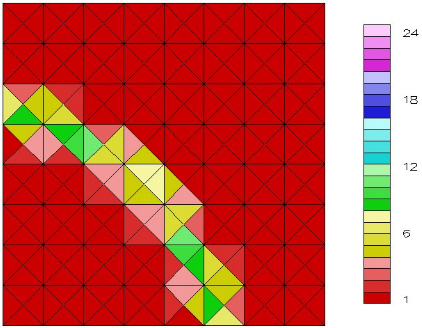
Adaptive Grid Refinement

- *h*-adaptive finite elements improve the accuracy by selectively subdividing elements to reduce the element size *h*
- *p*-adaptive finite elements improve the accuracy by increasing the polynomial degree, *p*, on selected elements
- *hp*-adaptive finite elements do both



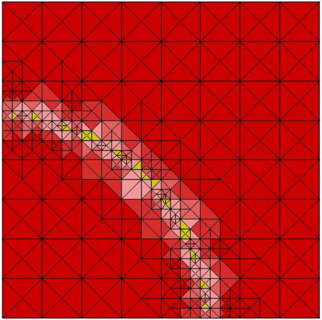
Adaptive Grid Refinement

- *h*-adaptive finite elements improve the accuracy by selectively subdividing elements to reduce the element size *h*
- *p*-adaptive finite elements improve the accuracy by increasing the polynomial degree, *p*, on selected elements
- *hp*-adaptive finite elements do both



Adaptive Grid Refinement

- *h*-adaptive finite elements improve the accuracy by selectively subdividing elements to reduce the element size *h*
- *p*-adaptive finite elements improve the accuracy by increasing the polynomial degree, *p*, on selected elements
- *hp*-adaptive finite elements do both



a priori error bounds

- (Babuška & Suri, 1987) If h and p are uniform and u is in the Sobolov space H^m

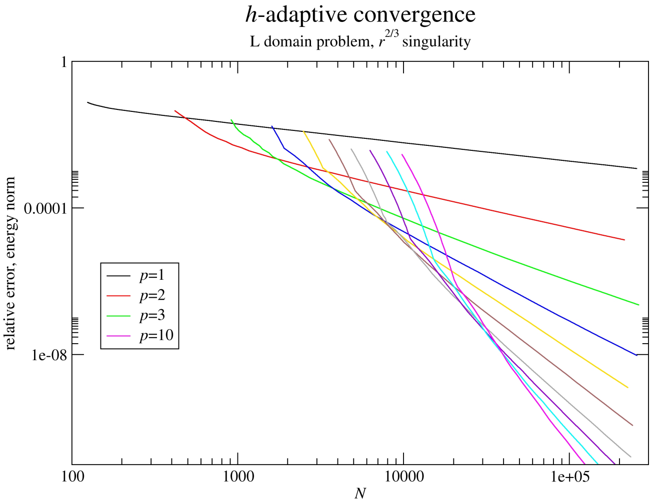
$$||u - u_{hp}||_{H^1} \leq C \frac{h^{\mu}}{p^{(m-1)}} ||u||_{H^m}$$

  where $\mu = \min(p, m-1)$
- this suggests that, if the solution is sufficiently smooth, p refinement is better, and if not, h refinement is better
- (Guo & Babuška, 1986) Convergence is exponential in the number of degrees of freedom, N

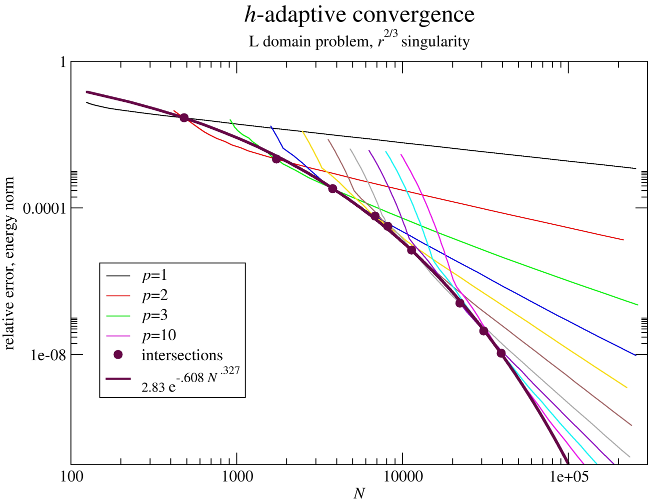
$$||u - u_{hp}|| \leq Ce^{-aN^b}$$

- in 2D, b is $1/3$

Exponenetial convergence



Exponential convergence



hp-Adaptive Methods

- Adaptive finite element methods use *a posteriori* local error indicators to determine *where* the grid should be refined
- But how do you determine *how* it should be refined?
  - by h?
  - by p?
  - some combination?
- Many *hp-adaptive strategies* have been proposed over the years to answer this question
- We have implemented these strategies in the finite element code PHAML and performed an extensive experiment to examine the performance of different strategies under different situations

a posteriori error indicators and estimates

- computable error indicators (or estimates) are used to determine which elements should be refined
- local Neumann error indicator: for an element, T_i, of degree p, use the p-hierarchical basis functions of exact degree $p+1$ to solve

$$Le_i = r := f - Lu_{hp} \quad \text{in } T_i$$
$$\frac{\partial e_i}{\partial n} = \left[\frac{\partial u_{hp}}{\partial n}\right] \quad \text{on } \partial T_i$$

where $Lu = f$ is the PDE, and $[\cdots]$ is the jump in the normal derivative across element boundaries

a posteriori error indicators and estimates

- $\eta_i = ||e_i||$ estimates the error over T_i
- $\eta = \sqrt{\sum \eta_i^2}$ estimates the global error
- note that η_i also estimates the amount of change in the solution if T_i was to be p-refined

hp-Adaptive algorithm

given an error tolerance, τ
begin with a very coarse grid in h with small p
discretize and solve on the coarse grid
loop
  compute η_i and η
  if $\eta < \tau$ **exit**
  mark elements with $\eta_i > \tau/\sqrt{N_{elem}}$ for refinement
  determine if marked elements should be refined by h or p
  refine marked elements
  discretize and solve on the current grid
end loop

- some strategies dictate a different algorithm
- to observe convergence, use $\tau = .1, .05, .025, .01, \ldots, 10^{-8}$

13 *hp*-Adaptive Strategies

methods for determining how an element should be refined

hp-Adaptive Strategies

- 1. Use of *a priori* knowledge (APRIORI)
  - Ainsworth & Senior, 1999
  - if there is *a priori* knowledge about the solution, use it
  - use h refinement at singularities and other trouble spots
  - use p refinement where the solution is smooth
- 2. Ratio of prior two p error estimates (PRIOR2P)
  - Süli, Houston & Schwab, 2000
  - for an element of degree p, compute error estimates η_{p-1} and η_{p-2} of the approximate solutions of lower degree
  - using the ratio of these η's and the *a priori* error bound

$$m \approx 1 - \frac{\log(\eta_{p-1}/\eta_{p-2})}{\log((p-1)/(p-2))}$$

hp-Adaptive Strategies

- 3. Type parameter (TYPEPARAM)
  - Gui & Babuška, 1986
  - directly use a ratio of error estimates, $R = \frac{\eta_p}{\eta_{p-1}}$
  - define a "type parameter", $0 \leq \gamma < \infty$, e.g. $\gamma = 0.3$
  - use h refinement if $R > \gamma$ and p refinement if $R < \gamma$
- 4. Convergence of next three p error estimates (NEXT3P)
  - Ainsworth & Senior, 1997
  - for an element of degree p, compute three error estimates based on spaces of degree $p+1$, $p+2$, and $p+3$
  - fit the three data points to the *a priori* error estimate to determine the three unknown constants in it, one of which is the smoothness m

hp-Adaptive Strategies

- 5. Texas 3 Step (T3S)
  - Oden & Patra, 1995
  - 1. perform uniform h refinement to get starting grid
  - 2. perform adaptive h refinement to reduce error part way
    - determine number of refinements by $n_e = (\eta_i^2 N_I/(\gamma\tau))^{1/\min(p+1,m)}$
  - 3. perform adaptive p refinement to reduce error to given tolerance
    - determine number of refinements by $p^{\mathrm{new}} = p(\eta_i\sqrt{N_I}/\tau)^{1/(m-1)}$
  - for high accuracy, use intermediate tolerances and repeat steps 2 and 3 until final tolerance is reached
- 6. Alternate h and p (ALTERNATE)
  - variant of Texas 3 Step
  - instead of computing how many times to refine an element, use our usual *hp*-adaptive algorithm
  - alternately refine by h and p to reduce the error to specific levels

hp-Adaptive Strategies

- 7. Nonlinear programming (NLP)
  - Patra & Gupta, 2001
  - formulate mesh design as an optimization problem
    - minimize total degrees of freedom subject to error less than tolerance, and other constraints (e.g. $p_i \geq 1$)
  - this leads to a mixed integer nonlinear program, which is NP-hard
    - allow real p and h, and round to the discrete values afterward
  - the solution gives new h and p for each element
- 8. Assume smooth and predict (SMOOTH_PRED)
  - Melenk & Wohlmuth, 2001
  - assume the solution is smooth, and predict what the error estimate should be under optimal convergence
  - perform h refinement if the actual error estimate is larger than the predicted error estimate, since that indicates the assumption of smoothness was violated, and p refinement otherwise

hp-Adaptive Strategies

- 9. Bigger of h and p error estimates (H&P_ERREST)
  - Schmidt & Siebert, 2000
  - one local *a posteriori* error indicator estimates how much the solution will change under p refinement by solving a local residual Neumann problem with the element p refined
  - another error indicator estimates how much the solution will change under h refinement by solving a local residual Dirichlet problem with the element h refined
  - compute both error indicators and select the type of refinement that will change the solution the most

hp-Adaptive Strategies

- 10. Decay rate of coefficients (COEF_DECAY)
  - Mavriplis, 1994
  - consider the coefficients of the expansion of the solution in the p-hierarchical basis
  - estimate the decay rate of the coefficients by a least squares fit of the last four to $ce^{-\sigma i}$
  - refine by p if $\sigma > 1$, and by h if $\sigma < 1$
- 11. Root test on coefficients (COEF_ROOT)
  - Houston, Senior & Süli, 2003
  - consider those same coefficients, a_i
  - estimate the regularity using a "root test"

$$m \approx \frac{\log((2p+1)(2a_p^2))}{2\log(p)} - \frac{1}{2}$$

hp-Adaptive Strategies

- 12. Reference solution, selection based on edges (REFSOLN_EDGE)
  - Demkowicz et al., 1989-2007
  - perform uniform h refinement and uniform p refinement and solve on the resulting mesh to get a reference solution $u_{h/2,p+1}$
  - stage a competition between edge p refinement and h refinements in which the children are assigned polynomial degrees that result in the same increase in degrees of freedom
  - for each possible refinement, determine the error decrease rate $(|u_{h/2,p+1} - w_{hp}|^2 - |u_{h/2,p+1} - w_{hp}^{\mathrm{new}}|^2)/(N_{\mathrm{new}} - N_{hp})$
    - w_{hp} is the projection based interpolant of $u_{h/2,p+1}$ on the original mesh, and w_{hp}^{new} is the interpolant on a competitor
  - basically choose the refinement with the largest error decrease rate, but there are several other subtleties

hp-Adaptive Strategies

- 13. Reference solution, selection based on elements (REFSOLN_ELEM)
  - Šolín et al., 2008
  - compute a reference solution $u_{h/2,p+1}$
  - candidate refinements for an element with degree p_i are
    - p refine to degree $p_i + 1$ and $p_i + 2$
    - h refine with all combinations of p_0, $p_0 + 1$, and $p_0 + 2$ where $p_0 = (p_i + 1)/2$
  - for each candidate, compute the H^1 norm projection error $\zeta_{\mathrm{candidate}} = ||u_{h/2,p+1} - w_{hp}||$
    - w_{hp} is the H^1 projection of $u_{h/2,p+1}$ onto the candidate refinement
  - choose the candidate that maximzes $(\log \zeta_i - \log \zeta_{\mathrm{candidate}})/(N_{\mathrm{candidate}} - N_i)$

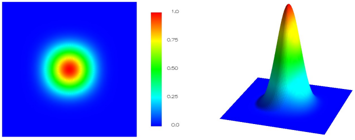
21 Test Problems

W.F. Mitchell, A Collection of 2D Elliptic Problems for Testing Adaptive Algorithms, NISTIR 7668, NIST, Gaithersburg, MD, 2010.

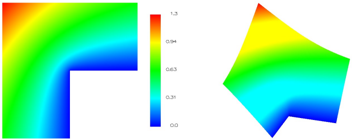
Test Problems

- Different equations
  - mostly Poisson, $u_{xx} + u_{yy} = f(x, y)$
  - one Helmholtz
  - one with first order terms
  - two with piecewise constant coefficients
  - one coupled system of two equations with mixed derivative term
- Boundary conditions
  - mostly Dirichlet
  - one with Neumann and mixed
- Domain
  - mostly unit square or (-1,1) square
  - some with reentrant corner or slit
- Exhibit a variety of difficulties
- Classified as easy, hard or singular

Test problems

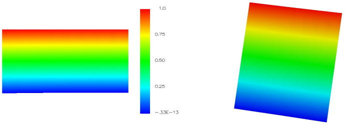


Analytic (polynomial)

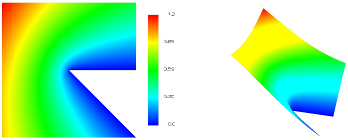


L-domain reentrant corner

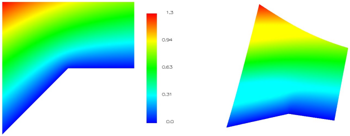
Test problems



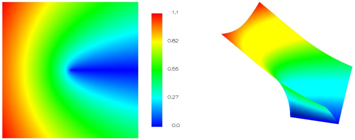
Nearly straight reentrant corner



Narrow angle reentrant corner

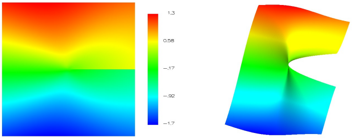


Wide angle reentrant corner

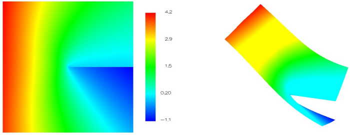


Slit

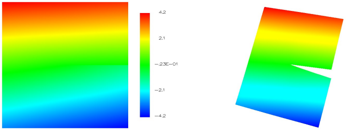
Test problems



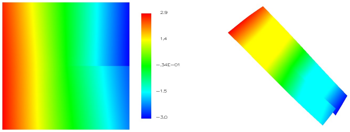
Linear elasticity, mode 1, u



Linear elasticity, mode 1, v

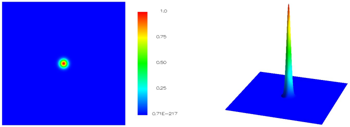


Linear elasticity, mode 2, u

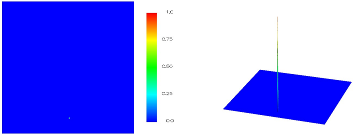


Linear elasticity, mode 2, v

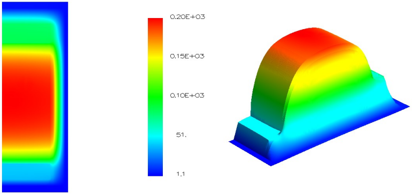
Test problems



Mild peak

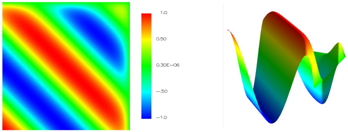


Sharp peak

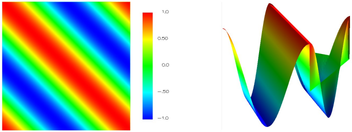


Battery

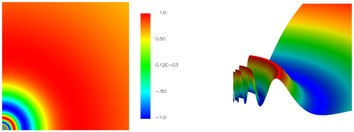
Test problems



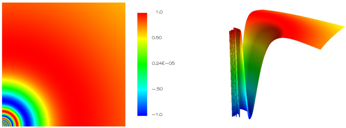
Mild boundary layer



Strong boundary layer

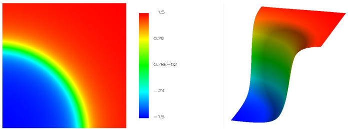


Mild oscillatory

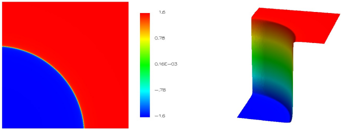


Strong oscillatory

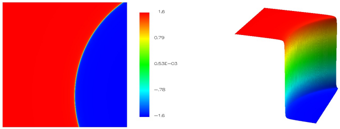
Test problems



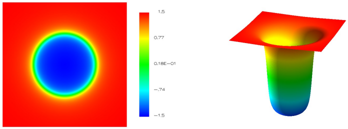
Mild wave front



Strong wave front

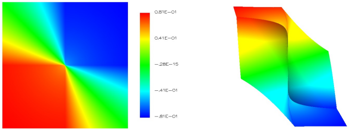


Asymmetric wave front

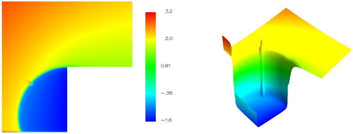


Singular well

Test problems



Intersecting interfaces

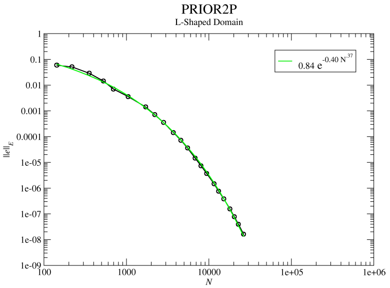


Multiple difficulties

Computational results

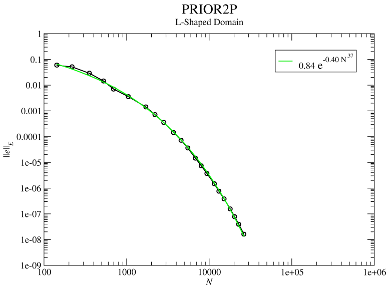
- Solve each problem with each strategy using a sequence of τ's
  - in most cases $\tau = .1,\ .05,\ .025,\ .01,\ .005, \ldots, 10^{-8}$
  - when the tolerance is met, record the number of degrees of freedom and energy norm of the error
- To get the convergence curve, compute the least squares fit of the form Ae^{-BN^C} to the data

Sample convergence curves

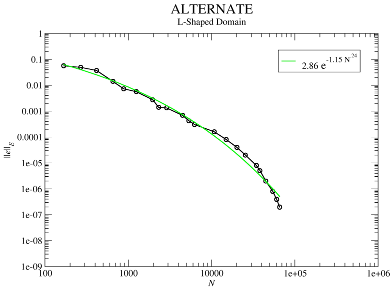


Most of the convergence data exhibit a very nice exponential convergence curve, although the exponent on N is not always this close to $1/3$

Sample convergence curves



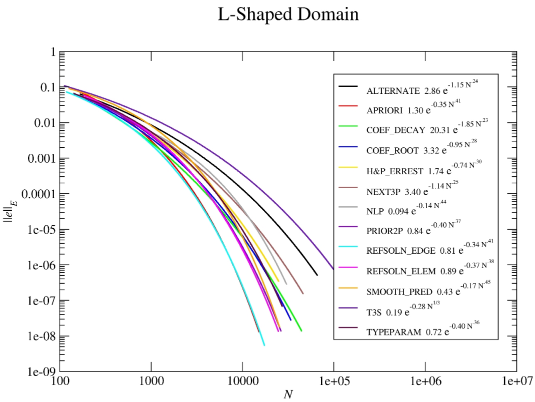
Most of the convergence data exhibit a very nice exponential convergence curve, although the exponent on N is not always this close to $1/3$



But some of them are a little sloppy

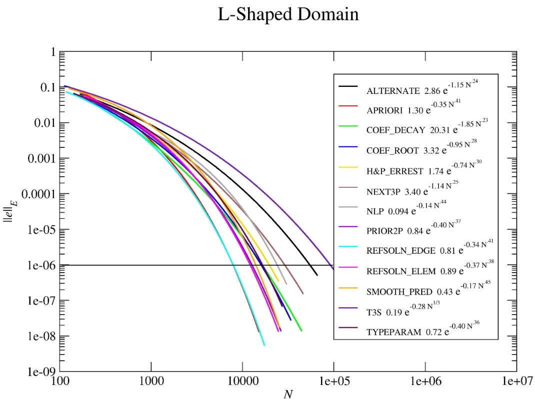
Ranking the strategies

1. For a given problem, consider the convergence curves of all strategies



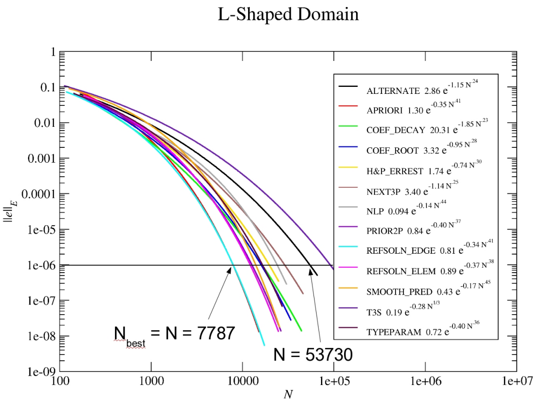
Ranking the strategies

2. Select an accuracy at which to rank the methods. For most problems, 10^{-2} for low accuracy, 10^{-6} for high accuracy



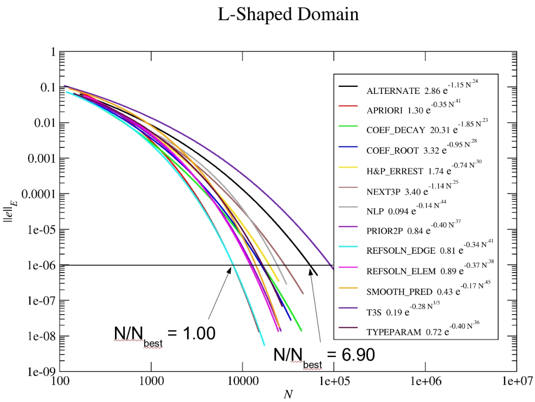
Ranking the strategies

3. Using the formula for the exponential curve, determine the N that gives the desired accuracy for each strategy



Ranking the strategies

4. Compute the factor by which N is larger than the smallest N



Ranking the strategies

5. Rank the strategies by this factor

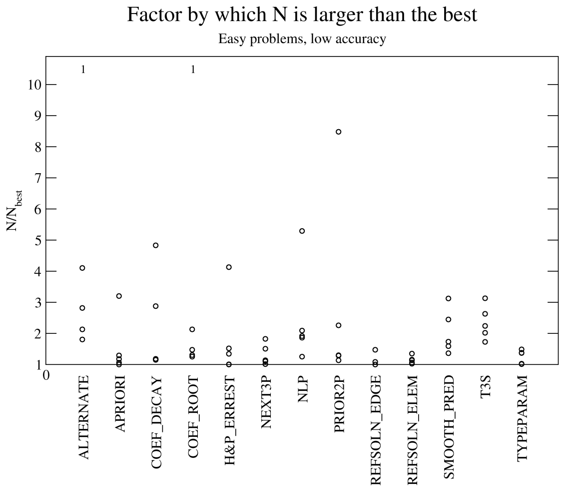
strategy	factor
APRIORI	1.00
REFSOLN_EDGE	1.00
REFSOLN_ELEM	1.54
PRIOR2P	1.61
SMOOTH_PRED	1.77
COEF_ROOT	2.03
COEF_DECAY	2.08
TYPEPARAM	2.09
H&P_ERREST	2.48
NLP	3.03
NEXT3P	3.67
ALTERNATE	6.90
T3S	11.55

L-shaped domain, high accuracy

Summary of results

- Group results by category and accuracy
  - easy problems, low accuracy
  - easy problems, high accuracy
  - hard problems, low accuracy
  - hard problems, high accuracy
  - singular problems, low accuracy
  - singular problems, high accuracy
  - all problems, low accuracy
  - all problems, high accuracy
- For each group, rank the strategies by the average of the N/N_{best} factors in that group
- When computing the averages, replace any factor that is greater than 10 with 10, so that a strategy is not disqualified by a single very bad case.

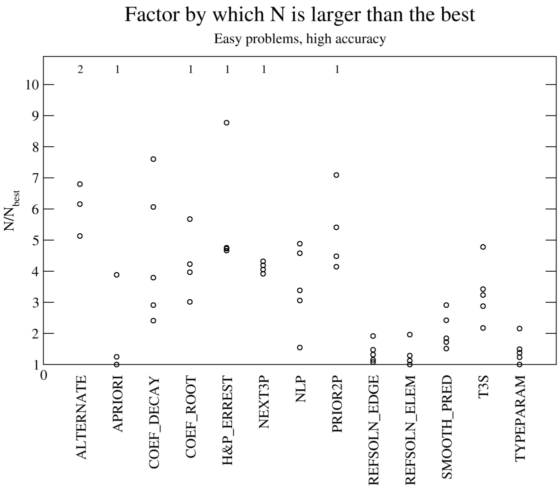
Easy problems, low accuracy



REFSOLN_EDGE	1.11
REFSOLN_ELEM	1.15
TYPEPARAM	1.25
NEXT3P	1.32
APRIORI	1.54
H&P_ERREST	1.80
SMOOTH_PRED	2.05
COEF_DECAY	2.24
T3S	2.35
NLP	2.48
PRIOR2P	2.89
COEF_ROOT	3.23
ALTERNATE	4.17

Average factor

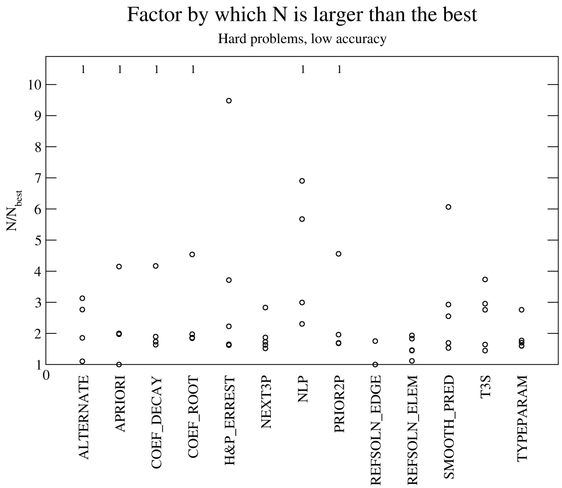
Easy problems, high accuracy



REFSOLN_ELEM	1.27
REFSOLN_EDGE	1.39
TYPEPARAM	1.45
SMOOTH_PRED	2.08
T3S	3.30
APRIORI	3.43
NLP	3.49
COEF_DECAY	4.55
NEXT3P	5.29
COEF_ROOT	5.38
PRIOR2P	6.22
H&P_ERREST	6.58
ALTERNATE	7.62

Average factor

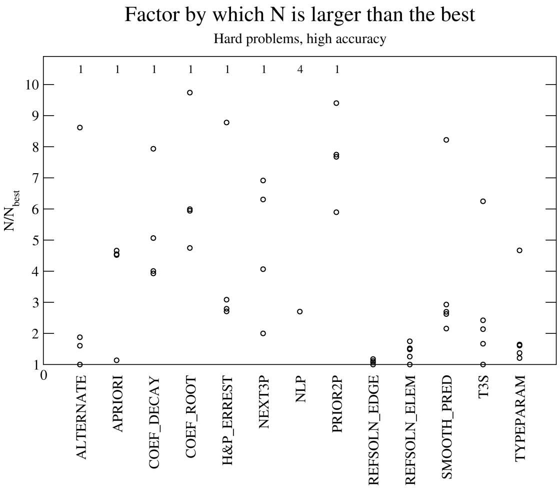
Hard problems, low accuracy



REFSOLN_EDGE	1.15
REFSOLN_ELEM	1.56
TYPEPARAM	1.89
NEXT3P	1.92
T3S	2.51
SMOOTH_PRED	2.95
H&P_ERREST	3.74
ALTERNATE	3.77
APRIORI	3.83
COEF_DECAY	3.89
PRIOR2P	3.98
COEF_ROOT	4.04
NLP	5.57

Average factor

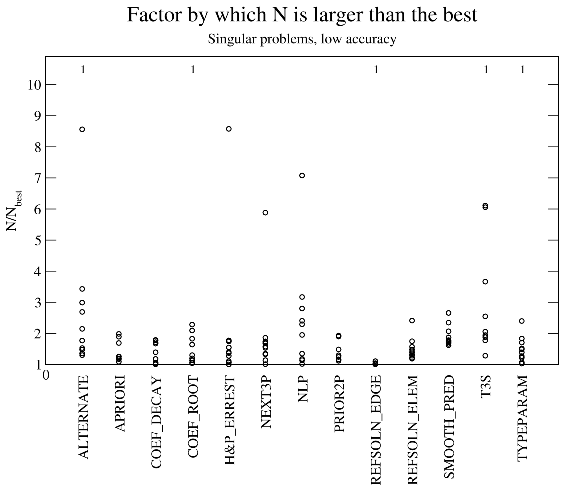
Hard problems, high accuracy



REFSOLN_EDGE	1.07
REFSOLN_ELEM	1.40
TYPEPARAM	2.10
T3S	2.69
SMOOTH_PRED	3.72
ALTERNATE	4.62
APRIORI	4.97
H&P_ERREST	5.47
NEXT3P	5.86
COEF_DECAY	6.19
COEF_ROOT	7.28
PRIOR2P	8.14
NLP	8.54

Average factor

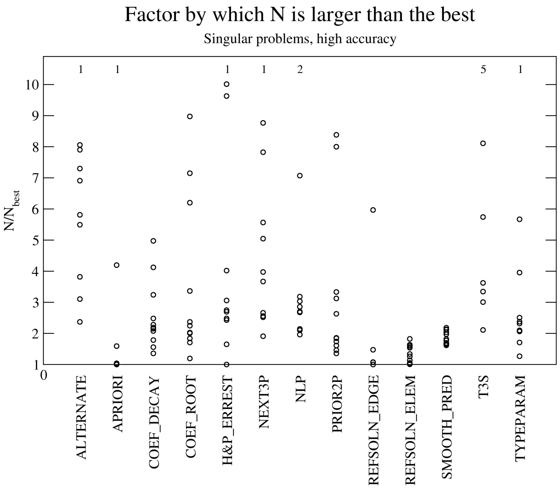
Singular problems, low accuracy



COEF_DECAY	1.33
APRIORI	1.37
REFSOLN_ELEM	1.47
PRIOR2P	1.54
REFSOLN_EDGE	1.83
NEXT3P	1.89
SMOOTH_PRED	1.92
H&P_ERREST	1.99
COEF_ROOT	2.24
TYPEPARAM	2.26
NLP	2.43
ALTERNATE	3.38
T3S	3.56

Average factor

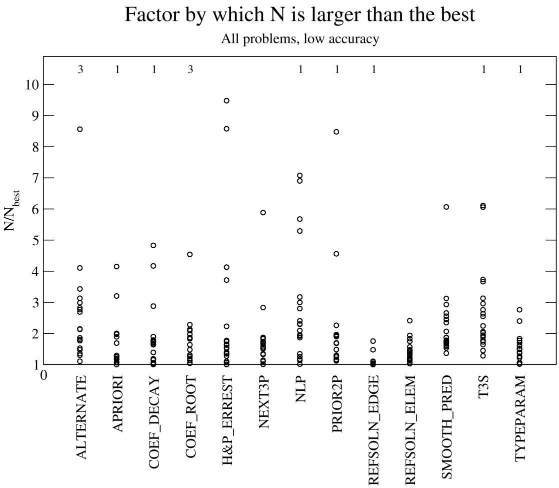
Singular problems, high accuracy



REFSOLN_ELEM	1.36
REFSOLN_EDGE	1.51
SMOOTH_PRED	1.88
APRIORI	2.17
COEF_DECAY	2.56
PRIOR2P	3.21
TYPEPARAM	3.27
COEF_ROOT	3.55
H&P_ERREST	3.86
NLP	4.34
NEXT3P	4.95
ALTERNATE	6.15
T3S	6.90

Average factor

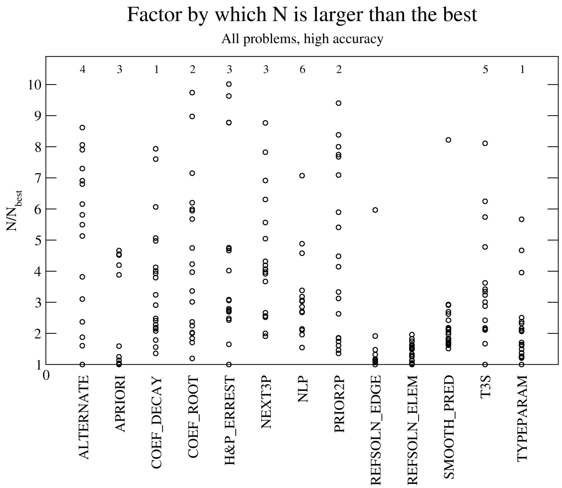
All problems, low accuracy



REFSOLN_ELEM	1.42
REFSOLN_EDGE	1.50
NEXT3P	1.76
TYPEPARAM	1.93
APRIORI	1.99
COEF_DECAY	2.15
SMOOTH_PRED	2.20
H&P_ERREST	2.36
PRIOR2P	2.44
COEF_ROOT	2.90
T3S	3.02
NLP	3.19
ALTERNATE	3.66

Average factor

All problems, high accuracy



REFSOLN_ELEM	1.35
REFSOLN_EDGE	1.38
SMOOTH_PRED	2.37
TYPEPARAM	2.56
APRIORI	3.14
COEF_DECAY	3.90
COEF_ROOT	4.87
H&P_ERREST	4.89
T3S	5.04
PRIOR2P	5.10
NLP	5.14
NEXT3P	5.25
ALTERNATE	6.13

Average factor

Timing

- Wall clock times for the mild wave problem
  - $\tau = 10^{-8}$
  - time spent in grid refinement
  - total time to solution
- NOT a careful timing comparison
  - Just to give a rough idea
  - Take it with a grain of salt

strategy	refinement	total
ALTERNATE	15.8	45.4
APRIORI	18.1	68.6
COEF_DECAY	6.7	23.5
COEF_ROOT	11.0	37.2
H&P_ERREST	42.8	113.6
NEXT3P	119.2	163.2
NLP	2317.8	2923.2
PRIOR2P	20.2	72.4
REFSOLN_EDGE	587.9	1188.1
REFSOLN_ELEM	136.4	143.2
SMOOTH_PRED	11.1	33.1
T3S	15.1	47.3
TYPEPARAM	20.8	55.0

Conclusions

- REFSOLN_EDGE and REFSOLN_ELEM are the top two strategies in all but one category
  - Not always the top two for every individual problem
  - Perform very well on every problem except REFSOLN_EDGE on battery
  - Considerably more expensive than all other methods except NLP and NEXT3P
- SMOOTH_PRED and TYPEPARAM perform very well in all categories
  - in the top 5 in most categories
  - SMOOTH_PRED is especially good at high accuracy
  - TYPEPARAM is especially good on non-singular problems
  - 1 problem where SMOOTH_PRED did not perform well (sharp peak)
  - 5 cases where TYPEPARAM did not perform well
  - very inexpensive, requiring just a couple simple computations

Conclusions

- APRIORI is very good for:
  - problems with point singularities at known locations
  - most problems at low accuracy
  - not so good for high accuracy solution of non-singular problems with strong features
- NEXT3P is exceptional at low accuracy, but bad at high accuracy and rather expensive
- T3S does OK with non-singular problems, but poorly with singular problems
- COEF_DECAY, COEF_ROOT and PRIOR2P do pretty good at low accuracy and for singular problems, but not as good for high accuracy solution of non-singular problems
- NLP is extremely expensive and does not perform very well

Future work

- Additional strategies
  - Strategies that have come to my attention recently
    - Eibner & Melenk (2007)
  - Strategies that have come into existence recently
    - Bank & Hguyen (2011)
    - Buerg & Doerfler (2011)
    - Wihler (2011)
- Use lessons learned from this study to develop a better general purpose strategy
  - Combine parts of different strategies that work well

Publications

- W.F. Mitchell and M.A. McClain, *A Survery of hp-Adaptive Strategies for Elliptic Partial Differential Equations*, in Recent Advances in Computational and Applied Mathematics (T.E. Simos, ed.), Springer, 2011, pp. 227–258.
- W.F. Mitchell, *A Collection of 2D Elliptic Problems for Testing Adaptive Algorithms*, NISTIR 7668, 2010.
- W.F. Mitchell and M.A. McClain, *A Comparison of hp-Adaptive Strategies for Elliptic Partial Differential Equations Using Bisected Triangles*, NISTIR 7824, 2011. Submitted to ACM TOMS.

- available at `http://math.nist.gov/~WMitchell`