Meshfree methods applied to the numerical solution of partial and integral equations

Edward J. Kansa

Convergent Solutions

Topics of implementation interest

- Convergence theory and implementation
- Poor conditioning of systems of equations
- Optimal discretization
- Domain decomposition & preconditioners
- Better solvers-Improved truncated-SVD
- High precision arithmetic
- Variable shape parameters
- Front tracking examples

H-scheme and c-scheme combined: PDEs and boundary conditions

- MQ is a prewavelet (Buhmann & Chui)
- Write MQ as $\phi_j(\mathbf{x}) =[1 +\{(\mathbf{x}-\mathbf{x}_j)/c_j\}^2]^{\beta}$
- $\mathbf{x}_j$ is the translator
- c_j is the dilator, and
- $[1 +\{(\mathbf{x}-\mathbf{x}_j)/c_j\}^2]^{\beta}$ is rotationally invariant.
- β influences the shape of $\phi_j(\mathbf{x})$.

- MQ cannot be a prewavelet if c_j is uniformly constant. In addition, the rows of the coefficient matrix are nearly identical.

Theoretical convergence and implementation

- Maych (1992) showed MQ interpolation and derivative estimates converge as:
- $O(\lambda^{\mu - |m|})$ where $0 < \lambda < 1$, $\mu = (c/h)$, and m is the order of differentiation,
- $D^m = \{\partial^{m1}\partial^{m2}\ldots\partial^{mk}\}/\ \{\partial x_1^{m1}\partial x_2^{m2}\ldots\partial x_k^{mk}\}$,
- $|m| = |m1|+|m2|\ldots+|mk|$, $h = \sup_{i,j}||x_i - x_j||$ (mesh size)
- Higher order differentiation lessens the convergence rate, and integration increases the convergence rate.

Goal: Obtain the best accuracy with minimal CPU time

- For convergence, we want $(c/h) \to \infty$.

- The h-scheme: refine h, keep c fixed.

- The c-scheme: increase c, use coarse h distribution.

- The c-scheme is ideal and most efficient.

Recommend h-scheme practices

- Brute force fine h discretization is a throw-back to mesh-based FDM,FEM, or FVM.
- High gradient regions require fine h and flatter regions require coarse h.
- The local length scale is: $\ell = k\,|U|/\,|\nabla U|$,U is the unknown dependent variable, k is a constant.
- Implementation: adaptive, multi-level local refinement are standard well-known tools.

Domain decomposition: Divide and Conquer for the h-scheme

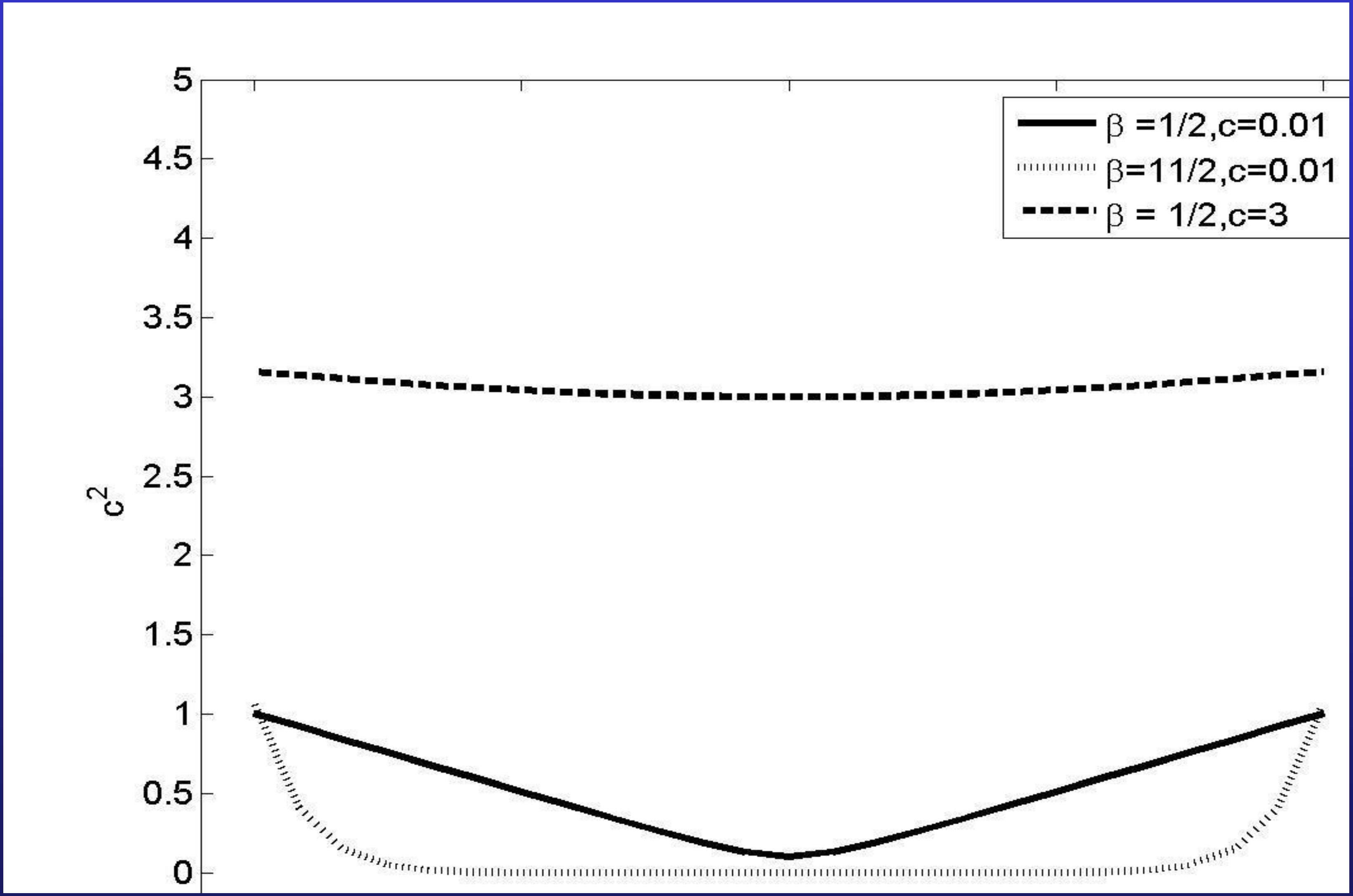
- Iterative Domain Decomposition: Parallel multilevel methods for elliptic PDEs (Smith, Bjorsted,Gropp) FEM
- Use overlapping or non-overlapping sub-domains
- For overlapping sub-domains, additive alternating Schwarz is fast, yields continuity of function and normal gradient.
- Smaller problems are better conditioned.
- Non overlapping methods yield higher continuity.
- Parallelization demonstrated by Ingber et al. for RBFs in 3D.

MQ shape is controlled by either c_j^2 or exponent, β

- ϕ_j should be "flat" near the data center, $\mathbf{x}_j$.
- Recommend using ½ integers β =3/2, 5/2, or 7/2; one can obtain analytic integrals for ϕ_j.

- Increasing c_j^2 makes ϕ_j "flatter".

Plots of 3 different MQ RBFs



FEM relies on preconditioners for large scale simulations.

- Even though FDM, FEM, and FVM have compact support, large systems of equations easily become ill-conditioned.
- Ill-conditioning can exist for RBFs PDE methods.
- Treatment of ill-conditioning will be presented in a separate presentation.

The c-scheme: advantages and disadvantages

- The c-scheme is very computationally efficient

- Unlike low order methods, the C^{∞} requires 100 – 1000 less resolution

- The disadvantage is the equation system becomes rapidly poorly-conditioned if limited to single or double precision.

Neumann Boundary Conditions and loss of Accuracy at the boundary

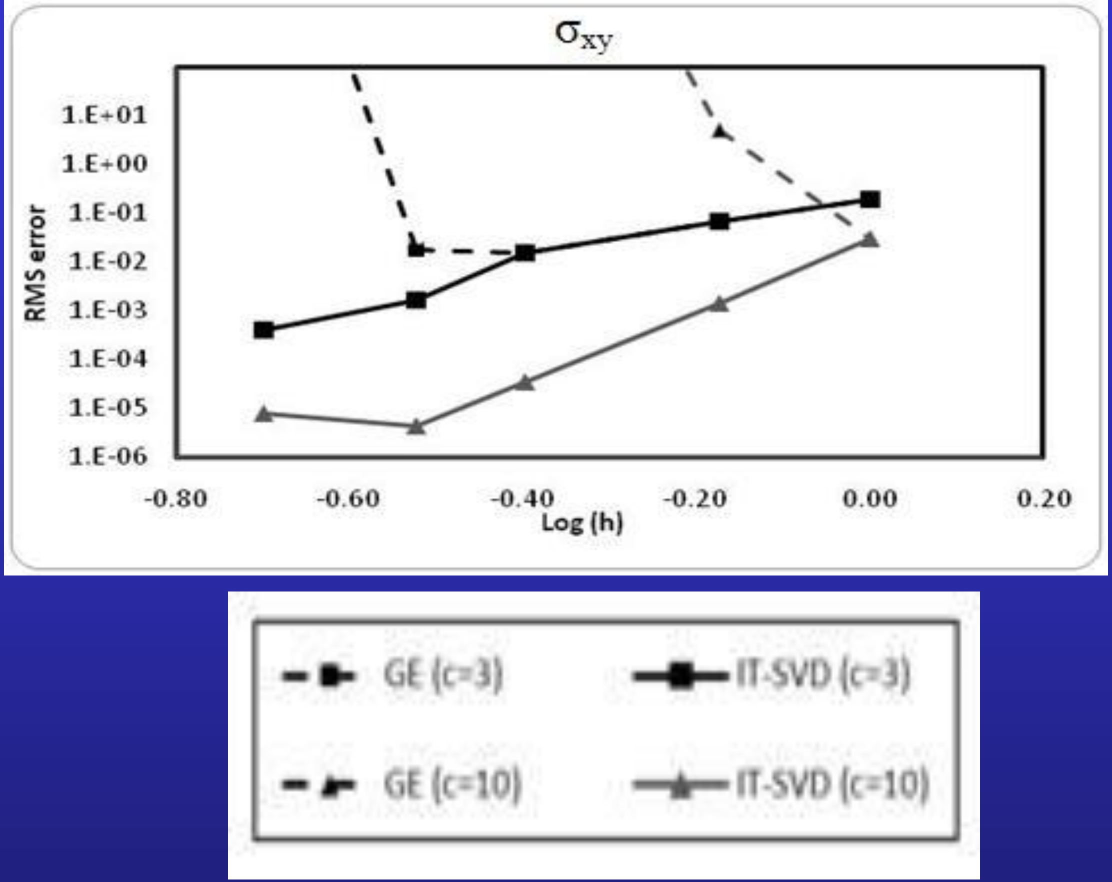
All numerical methods loose accuracy when derivatives are approximated.

MQ's rate of convergence is O($\lambda^{\eta-|\gamma|}$), where $\eta = c_j/h$ and γ is the order of spatial differentiation.

Remedy: Increase η so $\eta >> |\gamma|$.

Solid Mechanics problem

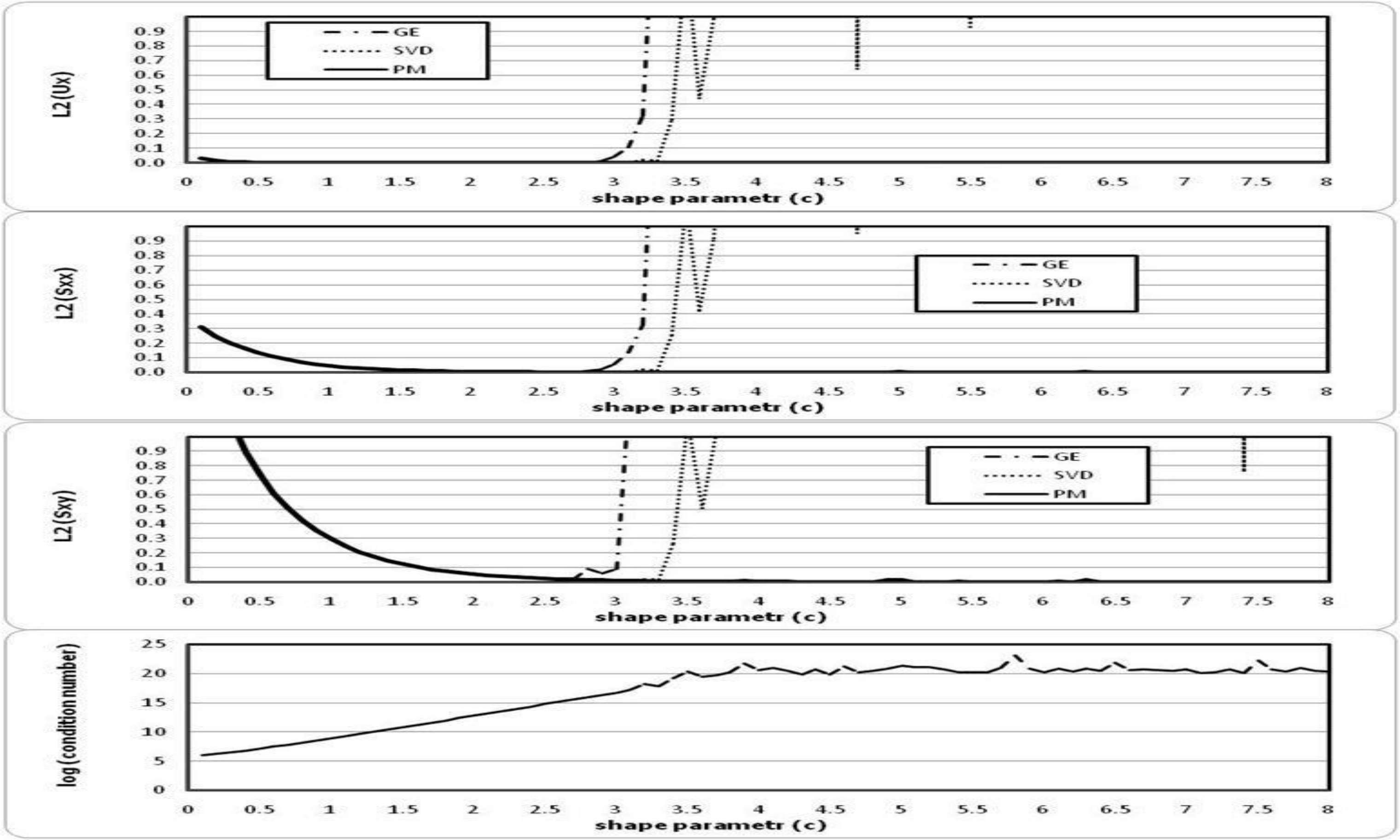
- u_x = (-P/6EI) (y-D/2)[(2+ν)y(y-D)] ;
- u_y = (PνL/2EI)(y-D/2)^2 {x=0, 0$\leq$ y $\leq$ D} $\partial\Omega_1$
- {x=L, 0$\leq$ y $\leq$ D} t_x = 0, t_y = (Py/2I)(y-D) $\partial\Omega_2$
- { 0 < x < D, y = 0, D} t_x =0, t_y = 0 $\partial\Omega_{2,4}$
- E = 1000, ν =1/3, L =12, D = 4, I= moment of inertia, P = applied force
- See Timoshenko and Goodier (1970).


σxy
RMS error
1.E+01
1.E+00
1.E-01
1.E-02
1.E-03
1.E-04
1.E-05
1.E-06
-0.80
-0.60
-0.40
-0.20
0.00
0.20
Log (h)
GE (c=3)
IT-SVD (c=3)
GE (c=10)
IT-SVD (c=10)

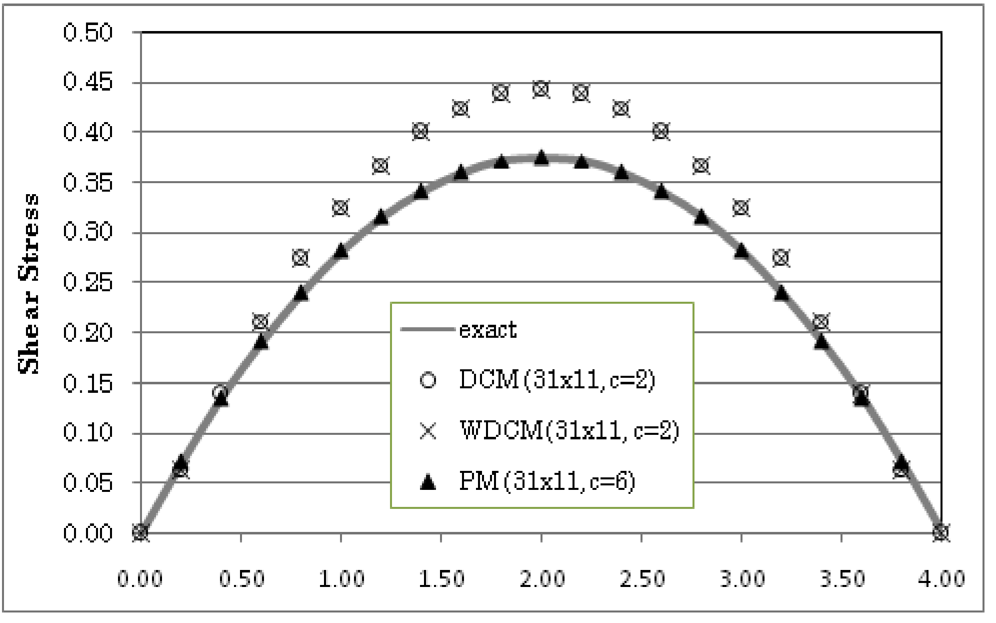
RMS errors with different solvers

Boundary Type	Neumann B.C.			Dirichlet B.C.		
Solver Method	GE	SVD	IT-SVD	GE	SVD	IT-SVD
u_x	1.48E-2	1.47E-2	5.82E-5	1.83E-4	8.38E-5	5.07E-6
u_y	1.27E-2	1.07E-2	3.35E-5	0.23E-4	1.25E-5	5.35E-7
σ_{xx}	4.34E-2	4.24E-2	8.38E-5	1.82E-3	9.13E-4	3.18E-5
σ_{yy}	3.78E-2	4.07E-2	8.82E-5	1.85E-2	1.03E-2	3.95E-4

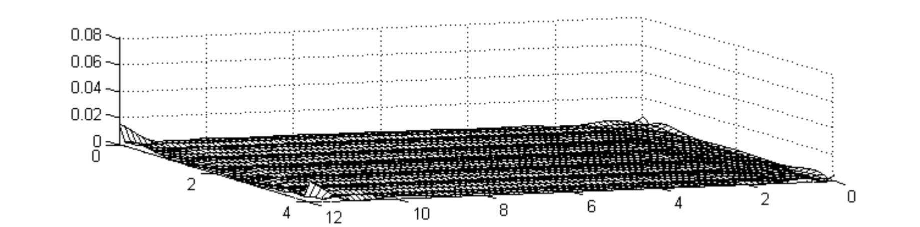
Dependency of L_2 errors on c (PM=IT-SVD)



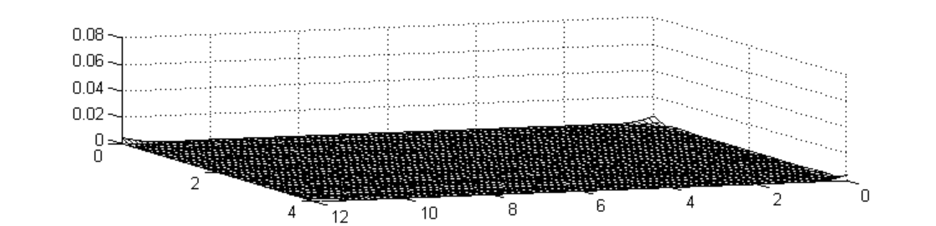
Shear stress at section x =L/2 of the beam with Neumann BC and PM=ITSVD



Neumann conditions: Good accuracy with IT-SVD scheme and large c^2_j



- Figure 4. Error distribution in stress field scattered data interpolation, (a) adaptive mesh refinement;



- (b) Adaptive shape parameter increment

Huang et al, EABE vol 31,pp614-624 (2007)

- They compared double & quadruple precision for the c- and h-schemes.

- For a fixed c & h, $t^{CPU}_{quad} = 40t^{CPU}_{double}$

- t^{CPU}_{quad} (c-scheme) = 1/565t^{CPU}_{double}(h-scheme).

- High accuracy & efficiency achieved with c-scheme.

How Accurate?

- Its accuracy is impossible to match by FEM or FDM.
- In an example solving Poisson equation, an accuracy of the order 10^{-16} is reached using a 20x20 grid.

To Make It Dramatic

- Assume that in an initial mesh, FEM/FDM can solve to an accuracy of 1%.
- Using a quadratic element or central difference, the error estimate is h^2
- To reach an accuracy of 10^{-16}, h needs to be refined 10^7 fold
- In a 3D problem, this means 10^{21} fold more degrees of freedom
- The full matrix is of the size 10^{42}
- The effort of solution could be 10^{63} fold
- If the original CPU is 0.01 sec, this requires 10^{54} years
- The age of universe is 2 x 10^{10} years

Variable c_j-Fornberg & Zeuv (2007)

- They chose $\varepsilon_j = 1/c_j = 1/c_{ave} d_j$, where d_j is the nearest neighbor distance at x_j.

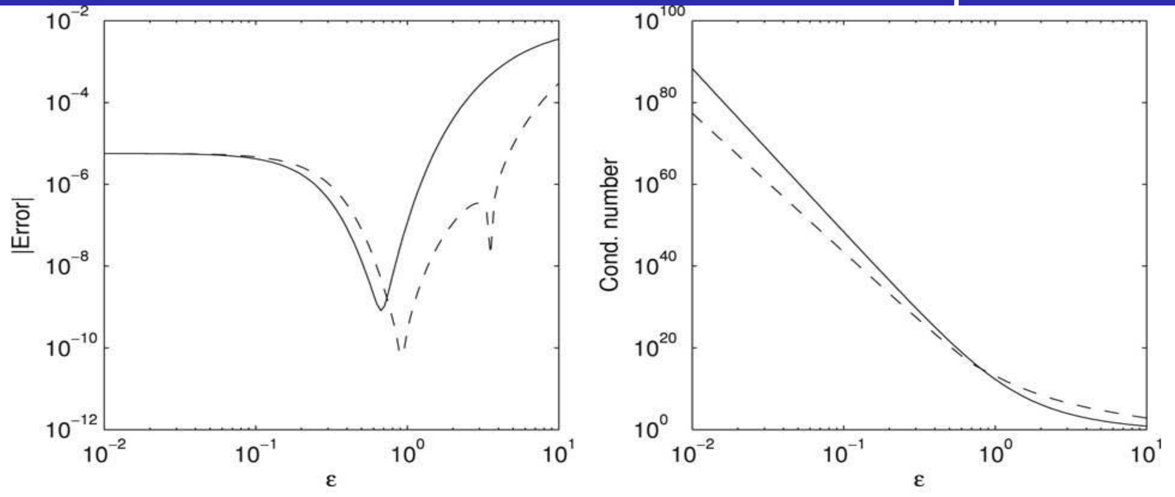


Fig. 7.4. Comparison of results when interpolating $f(x) = \frac{1}{1+x^2}$ on an equispaced grid with $n = 21$ points over $[-1, 1]$, using spatially constant ε (solid lines) and spatially variable ε_k (as described in the text; dashed lines) for (a) max norm error and (b) condition number for linear system.

Boundary condition implementation

- The PDE exists everywhere, but the boundary conditions to be unique if the problem is well posed

- Allow some points to be slightly outside of Ω

Implementation recommendations for RBF PDEs (9)- MQ shape parameters

Consider the MQ RBF

$\varphi_k(x)=[\ 1+(x-\xi_k)^2/c_k^2]^\beta\ \ (\beta \geq -1/2)$ (MQ)

Wertz, Kansa, Ling (2005) show:

1. Let $\beta \geq 5/2$; asysmpotically MQ is a high order polyharmonic spline

2. Let $(c_k^2)_{\partial\Omega} \geq 200(c_k^2)_{\Omega\backslash\partial\Omega}$

Comments on Boundary condition implementation and convergence

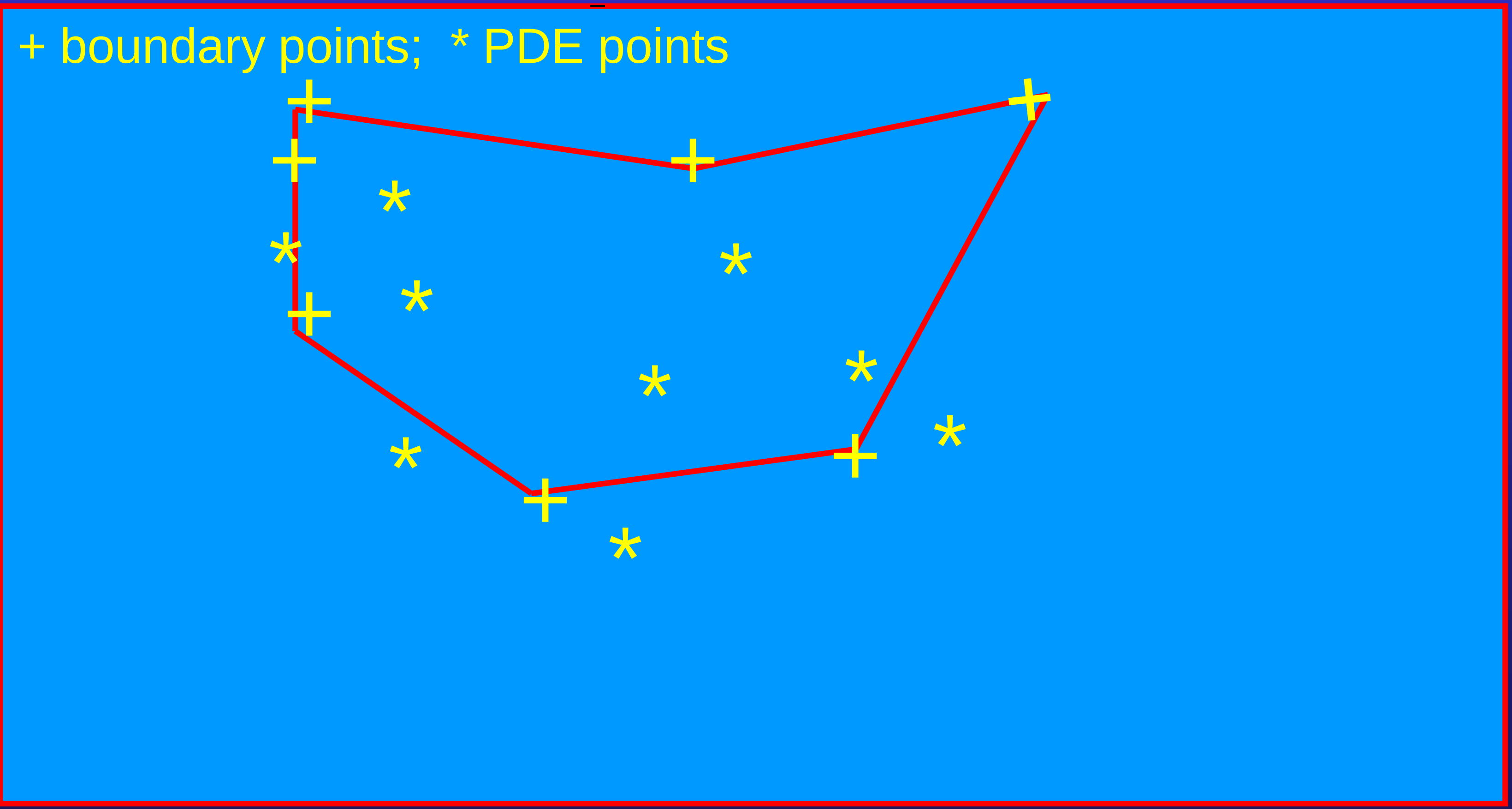
- Just using a equi-distributed set of data centers is not sufficient for accurate representation of Neumann BCs

- Specifying $-k\partial T/\partial n=g$ can be inaccurate if centers inside and outside $\partial\Omega$ are too widely separated

Fedoseyev et al.(2002)

- By extending the PDE domain to be slightly outside of the boundaries, they observed exponential convergence for 2D elliptic PDEs.

H-scheme- PDE exist everywhere in $\Re^d$, extend the domain outside of boundaries



Fornberg & Zuev, Comp.Math.Appl. (2007) Variable $\varepsilon_j = 1/c_j$ reduces cond.number, improves convergence

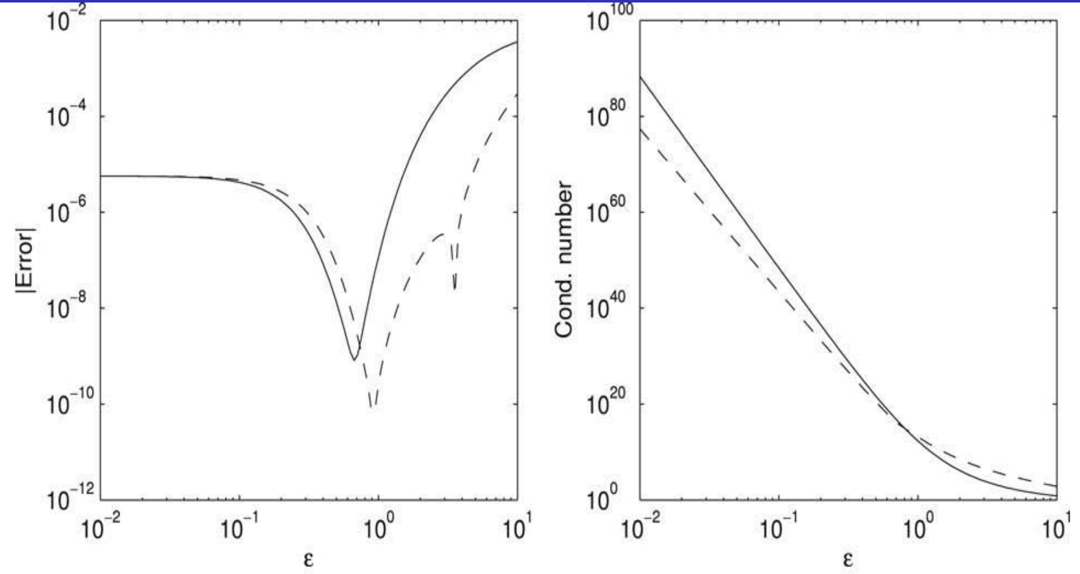


Fig. 7.4. Comparison of results when interpolating $f(x) = \frac{1}{1+x^2}$ on an equispaced grid with $n = 21$ points over $[-1, 1]$, using spatially constant ε (solid lines) and spatially variable ε_k (as described in the text; dashed lines) for (a) max norm error and (b) condition number for linear system.

Summary of Wertz study

- Using $\beta > ½$ produces more rapid convergence.
- Boundary conditions make the PDE unique (assuming well posedness), hence $(c_j^2)_{\partial\Omega} >> (c_j^2)_{\Omega\backslash\partial\Omega}$
- Permitting the (c_j^2) on both the $\partial\Omega$ and $\Omega\backslash\partial\Omega$ to oscillate reduces RMS errors more, perhaps producing better conditioning.

Front tracking is simple with meshless RBFs

- No complicated mesh cell divisions.
- No extremely fine time steps using above method.
- No need for artificial surface tension or
- viscosity.

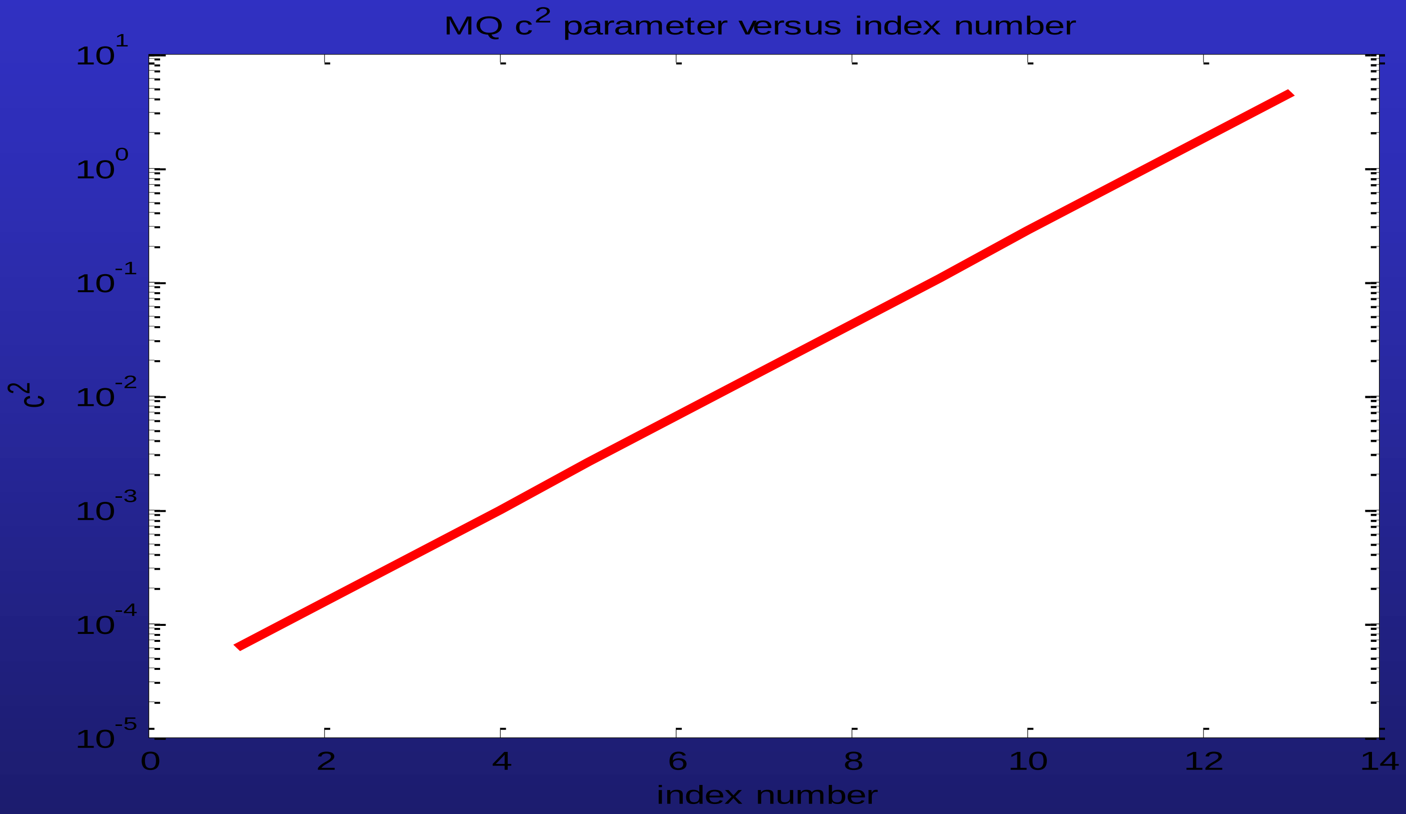
Sethian’s test of cosine front

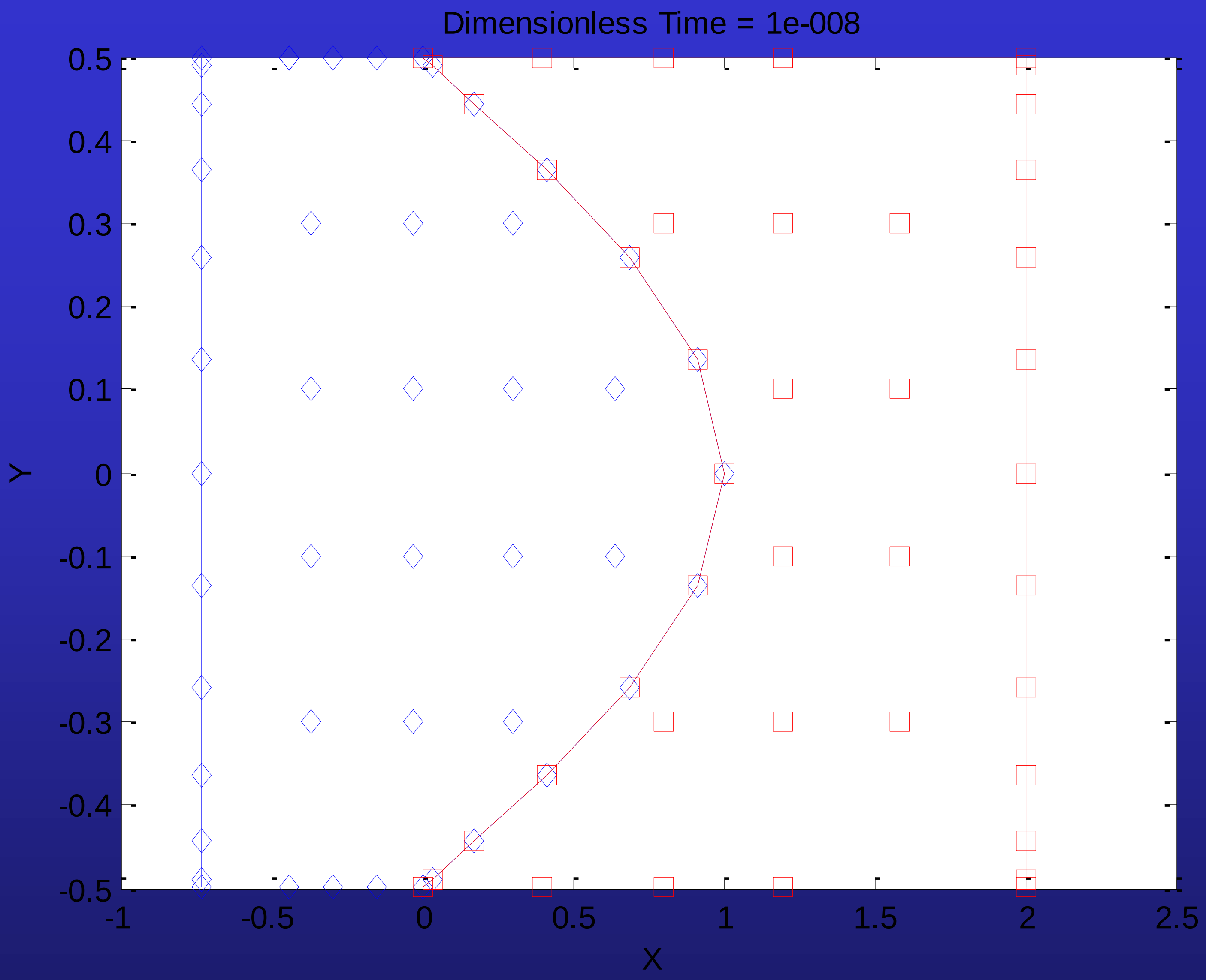
- At t=0, flame is a cosine front, separating burnt and unburnt gases.
- This front should develop a sharp cusps in the direction of the normal velocity.
- Conversely, a front should flatten when it faces in the opposite direction.
- The flame front moves by the jump conditions in the local normal direction.

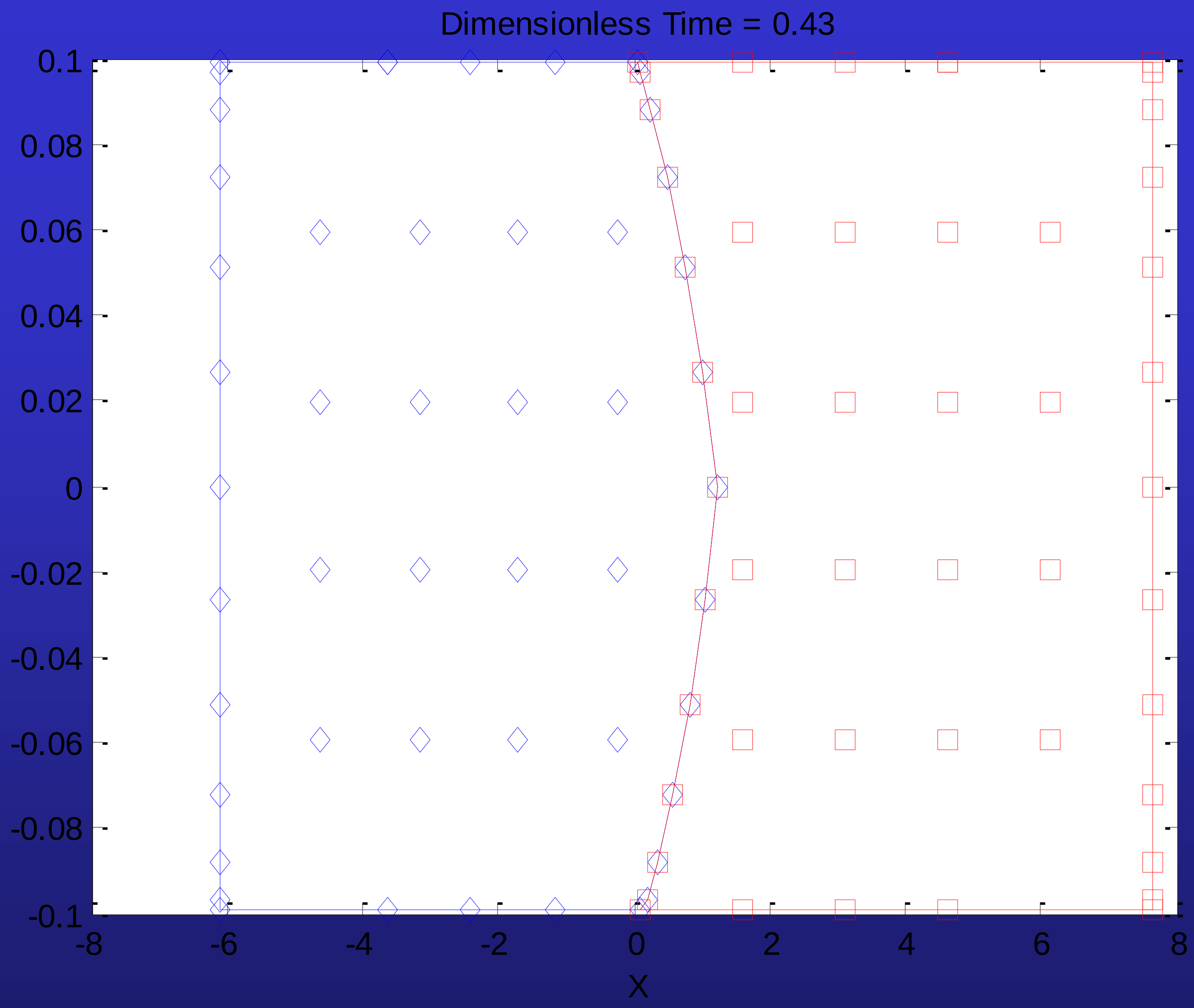
Front tracking is very hard with meshes

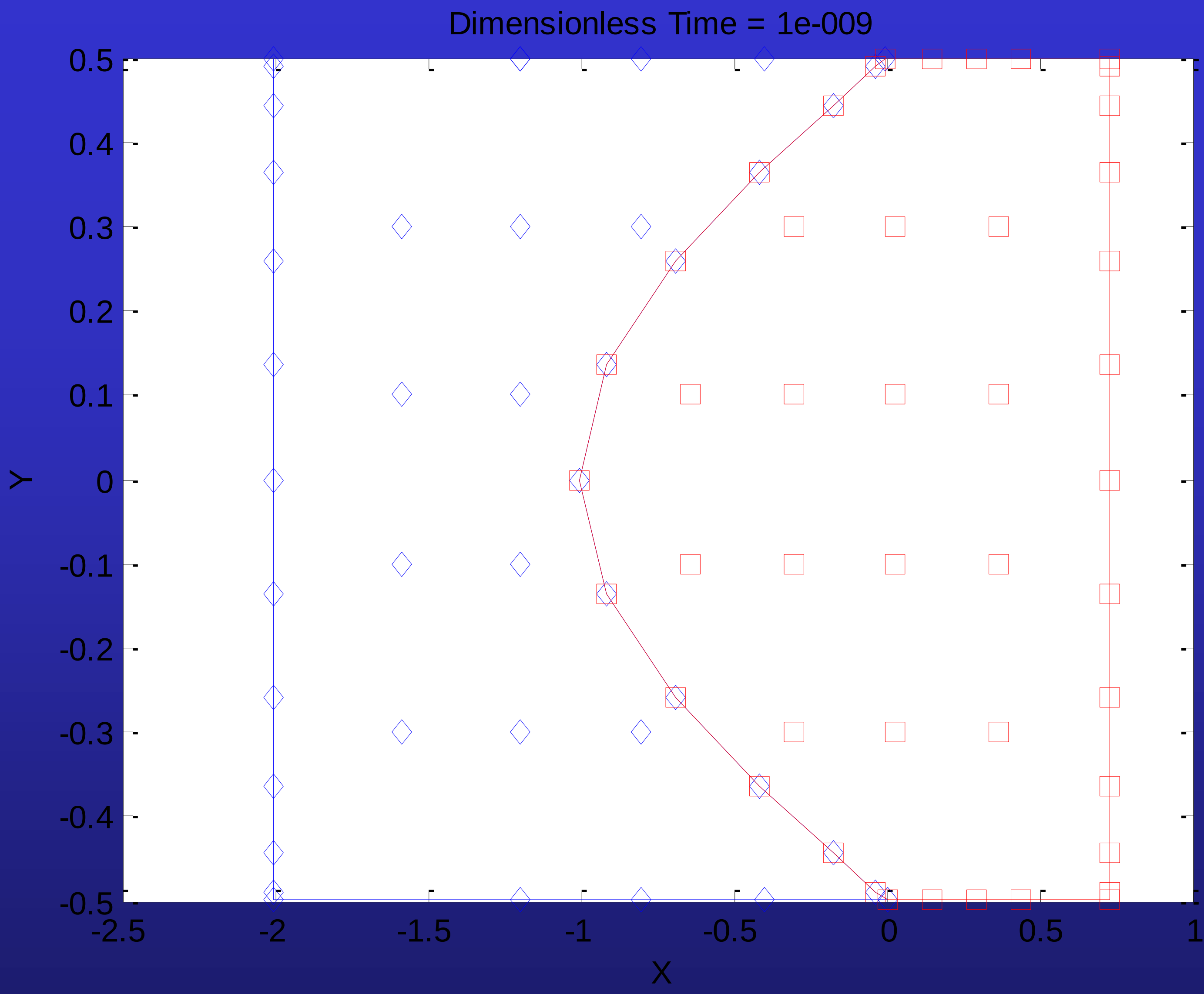
- Front capturing requires unphysical viscosity.
- Complicated problems of mesh unions and divisions as front moves in time.
- The tangential front is usually not a spline, artificial surface tension and viscosity are required for stability.

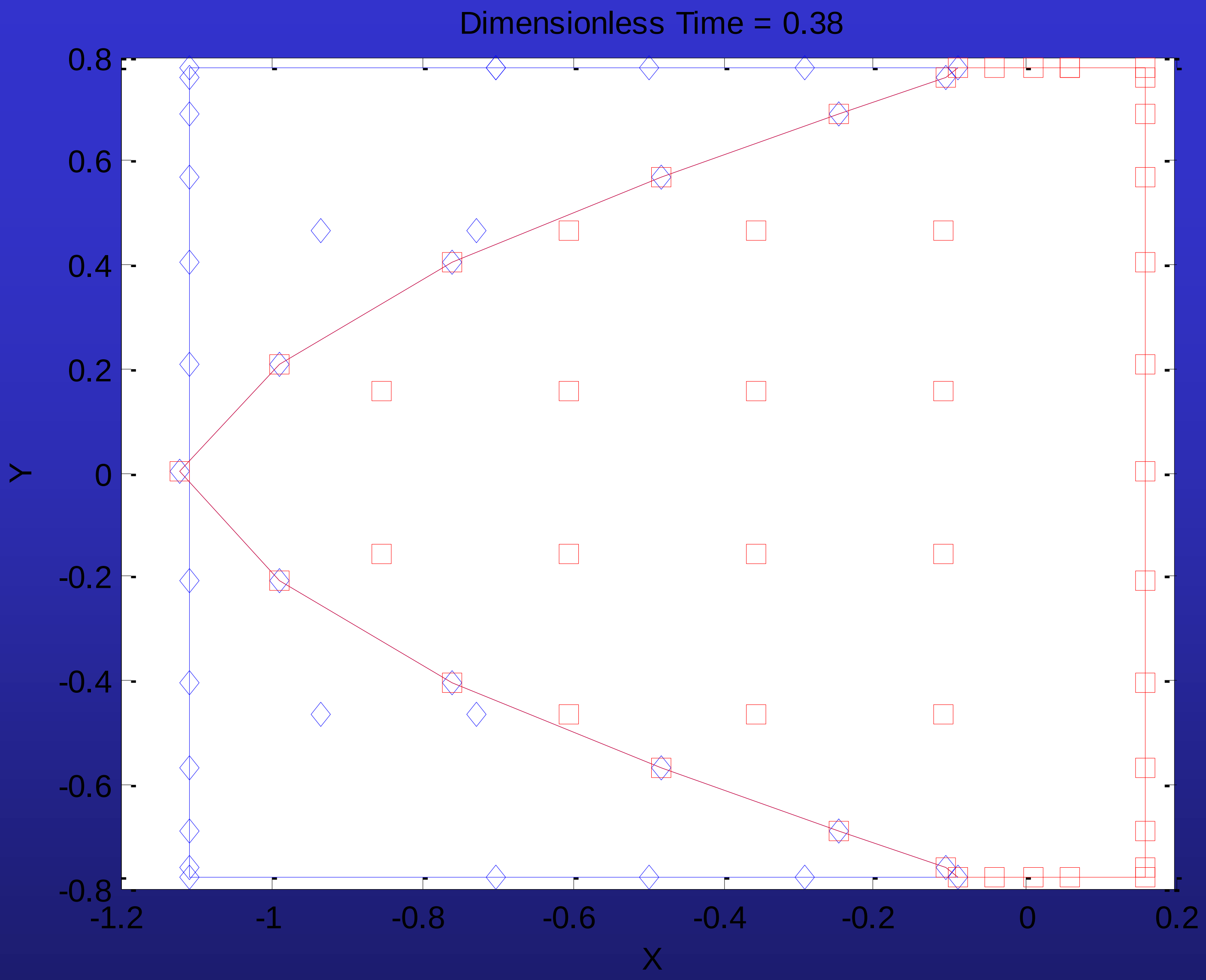
In 1990, Kansa showed the best performance with variable c_j^2 ,not a constant.



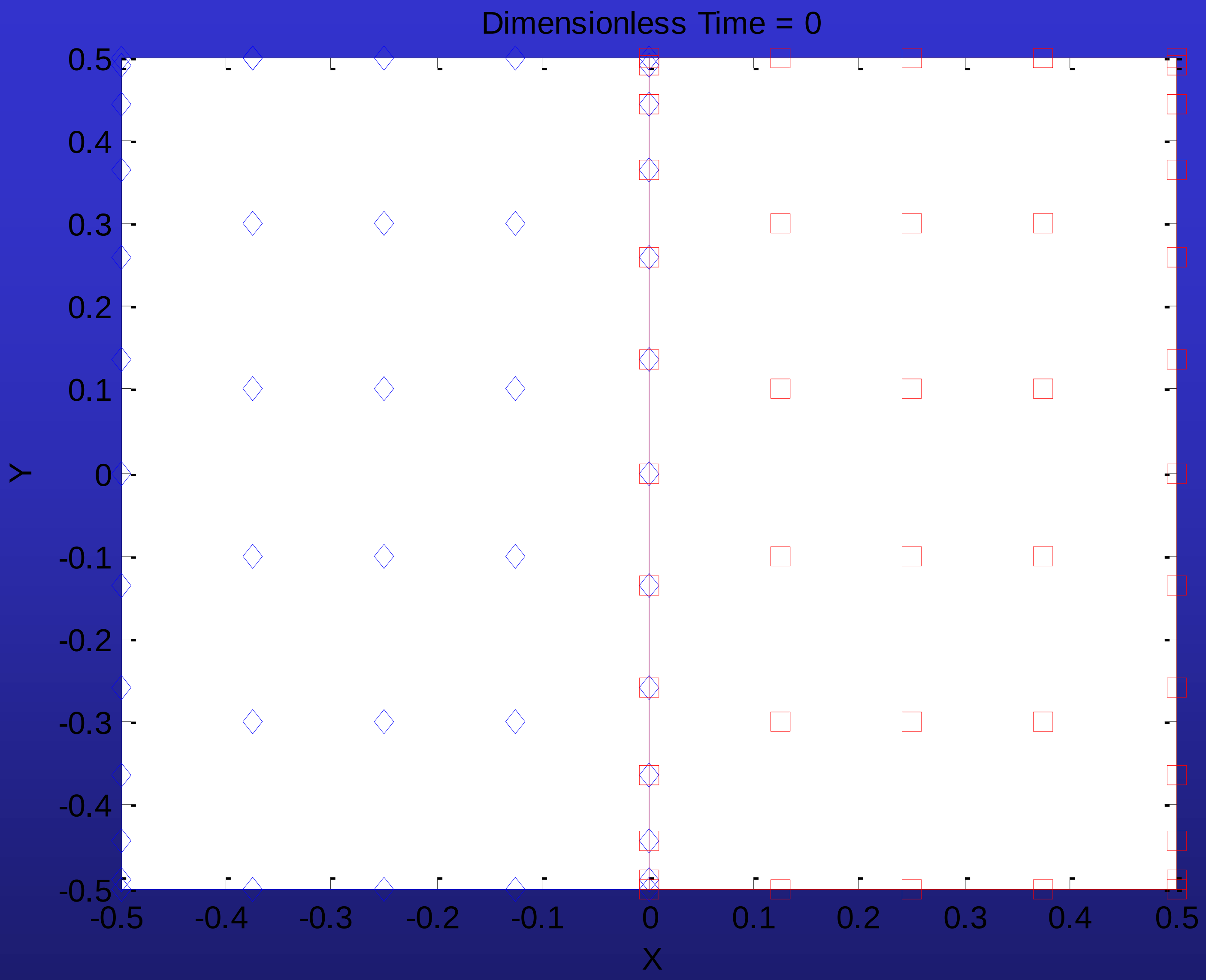

Dimensionless Time = 1e-008
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
Y
-1
-0.5
0
0.5
1
1.5
2
2.5
X

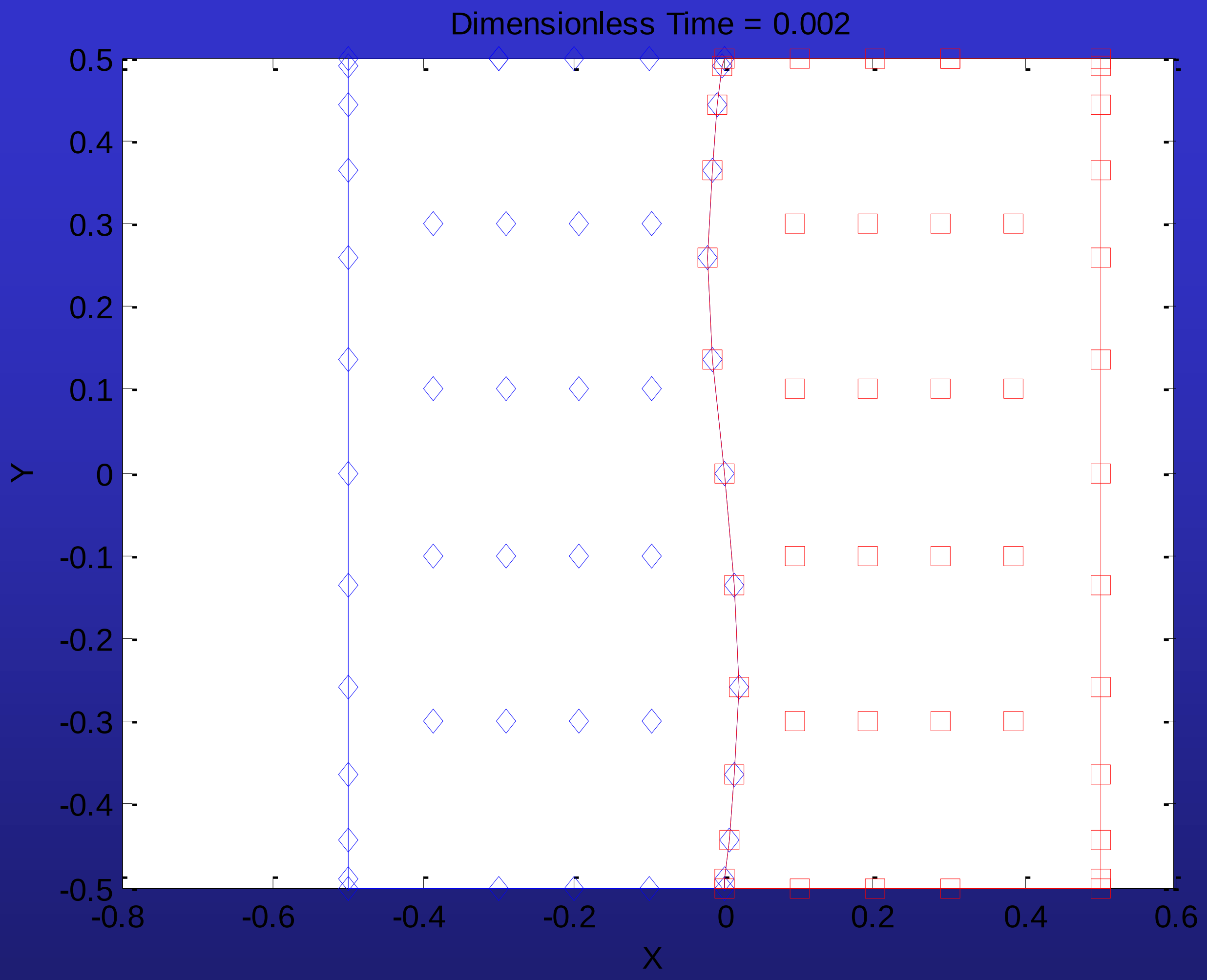

Dimensionless Time = 0.43
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
-8
-6
-4
-2
0
2
4
6
8
X

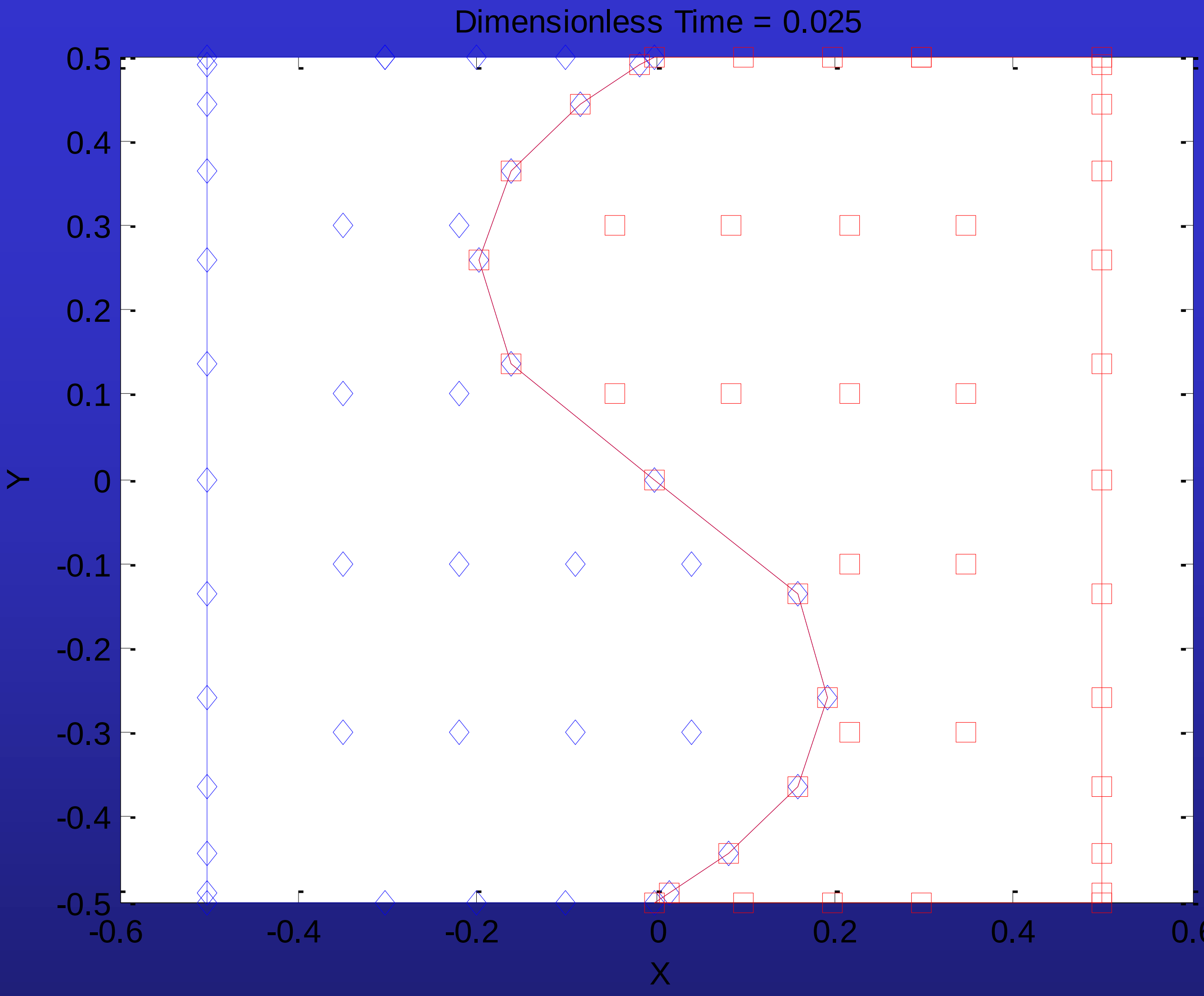

Dimensionless Time = 1e-009
Y
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
X

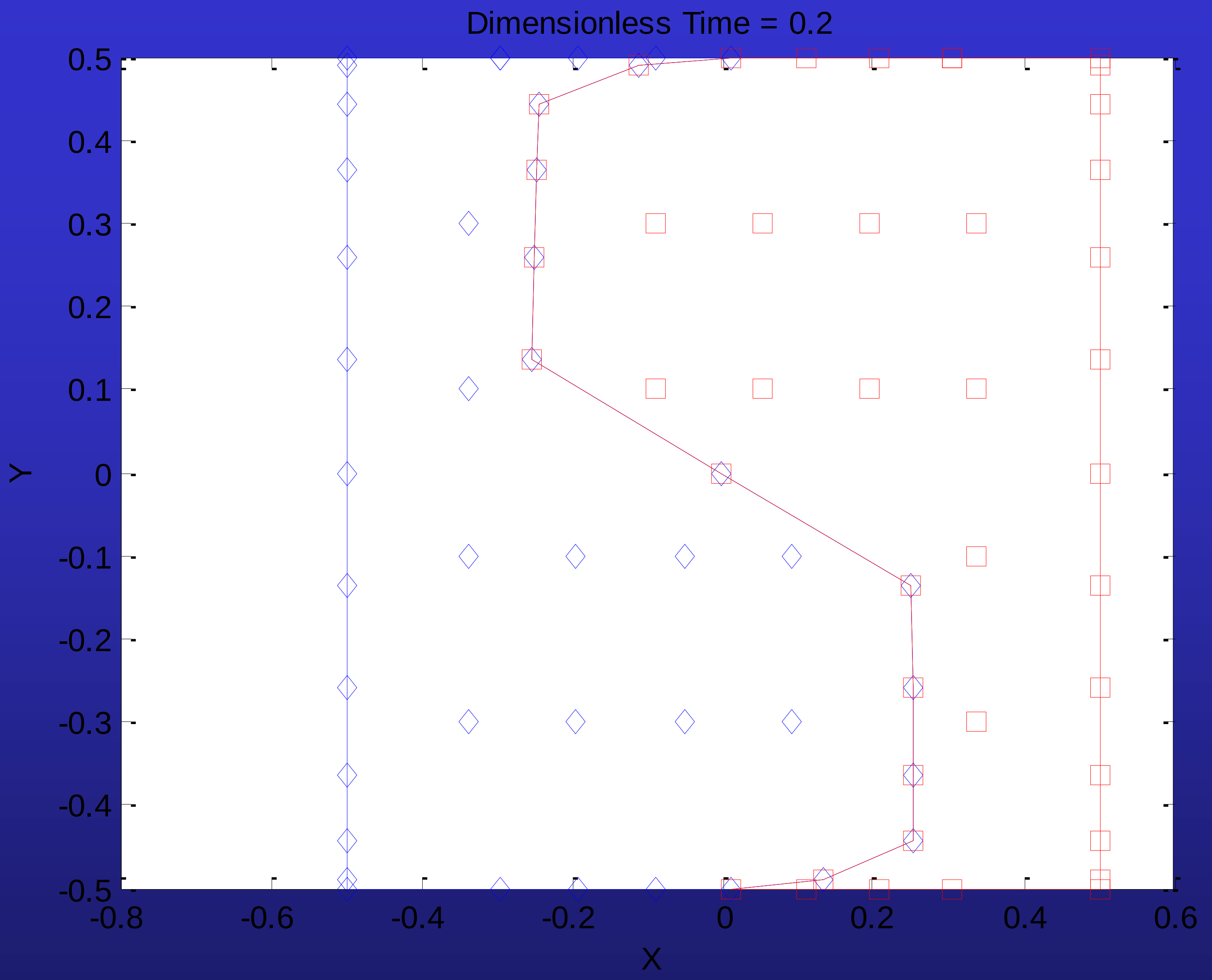

Dimensionless Time = 0.38
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
Y
-1.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
X

- 2D Vortical turbulent combustion
  - 2D infinitely periodic turbulent flame.
  - PDEs are hyperbolic, use exact time integration scheme, EABE vol.**31** 577–585 (2007).
  - Flame front is a discontinuous curve at which the flame speed is normal to flame front.
  - Two separate subdomains used: burnt and unburnt gases, jump conditions for flame propagation.

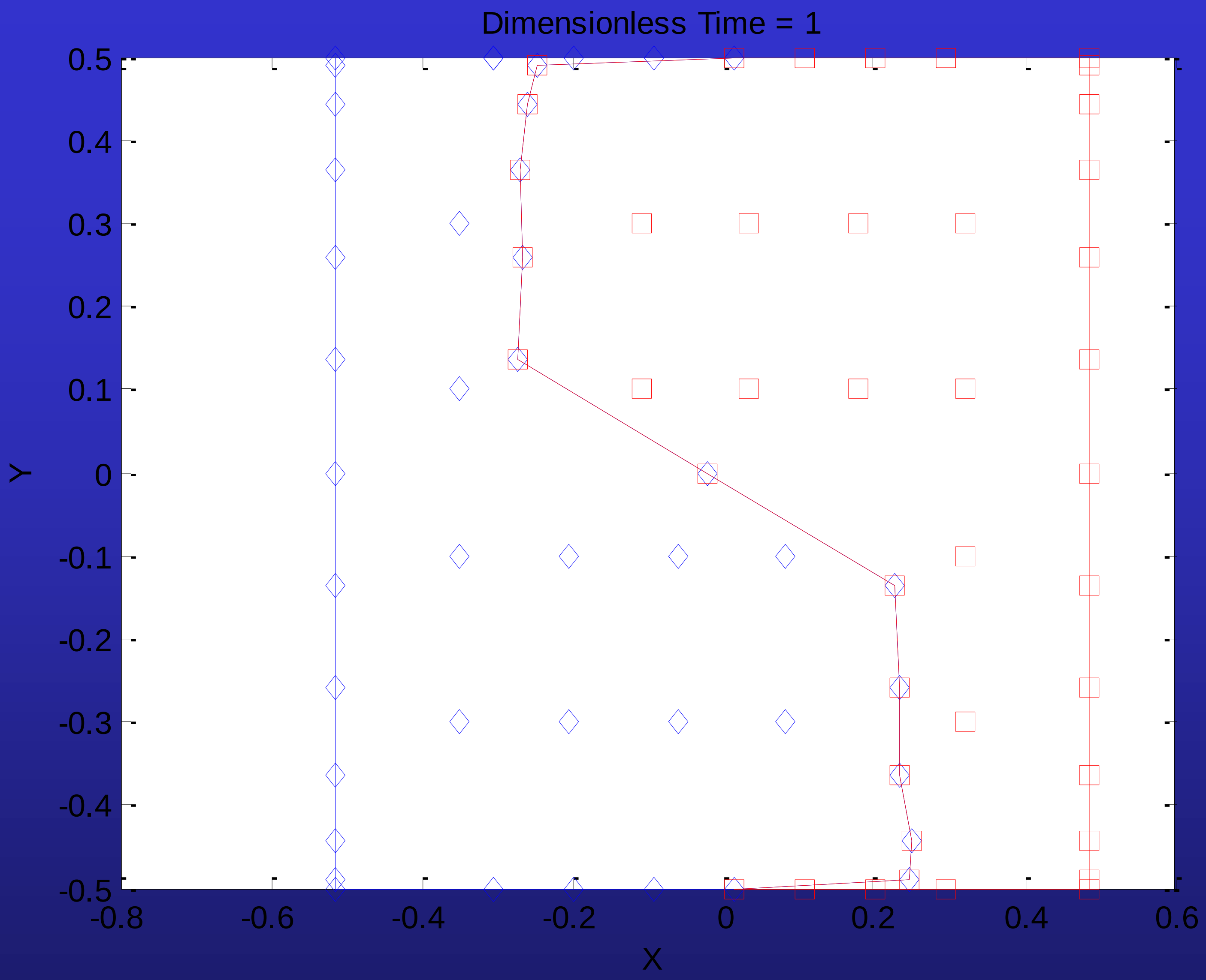

Dimensionless Time = 0
Y
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.5
-0.4
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
X


Dimensionless Time = 0.002
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
Y
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
X


Dimensionless Time = 0.025
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
Y
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
X



Dimensionless Time = 0.2
Y
X
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6


Dimensionless Time = 1
Y
X
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6

Turbulent flame propagation studies

- Traditional FDM required 14 hrs on a parallel computer to reach the goal time of 1.
- Time required for the RBF method to reach the goal time of 1 was 23 seconds on a PC.

Summary

- Use spatial refinement sparingly.
- The variable $c^2_j = |U| /|\nabla U|$ is more stable, accurate and better conditioned.
- The IT-SVD projects small singular values into the null space.
- Need to investigate Huang et al.'s claim that extended precision is indeed cost-effective in minimizing *total* CPU time.
- Hybrid combinations of domain decomposition, preconditioning, variable c's, IT-SVD, & extended precision need to be examined.

Efficiency of meshless MQ-RBFs versus traditional, long established FDM,FEM, & FVM

- CPU time (FDM,FEM, FVM)/discretization pt << CPU time(meshless MQ)/discretization pt
- END OF STORY – NO
- BOTTOM LINE – total CPU time to solve a PDE problem, t_{CPU}(RBF) <<t_{CPU}(FEM,FDM,FVM).