Adaptive Radial Basis Function Methods with Residual Subsampling Technique for Interpolation and Collocation Problems

Alfa Heryudono   Tobin Driscoll

Department of Mathematical Sciences
University of Delaware

SIAM Annual Meeting July 13 2006
Boston, Massachusetts

Given data at nodes $\mathbf{x}_1, ..., \mathbf{x}_N$ in d dimensions, the basic form for an RBF approximation is

$$F(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j \phi(\epsilon_j ||\mathbf{x} - \mathbf{x}_j||),$$

where $|| \cdot ||$ denotes the Euclidean distance between two points and $\phi(r) = \sqrt{1 + r^2}$ is defined for $r \geq 0$.

$$f_i = f(\mathbf{x}_i)$$

$$\Longrightarrow$$

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_N \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}$$

where $a_{ij} = \phi(\epsilon_j ||\mathbf{x}_i - \mathbf{x}_j||)$. Nonsingularity of A is guaranteed for many choices of ϕ with mild restrictions and constant shape parameters ϵ_j.

Advantages of RBF methods

- No need for a mesh / triangulation.
- Simple implementation and dimension independence.
- No staircasing / polygonization for boundaries.
- Depending on chosen RBFs, high-order/spectral convergence can be achieved.
- Easy to implement derivatives and boundary conditions.

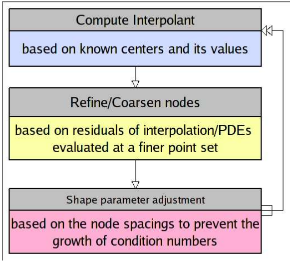
Challenges using RBF methods

- As the number of centers grows, the method needs to solve a relatively large algebraic system
- The matrix is full (except for compactly supported RBF).
- Choosing nodes and shape parameters.
- Ill-conditioning usually makes spectral convergence difficult to achieve.

Problems involve

- geometry
- steep gradients
- corners
- topological changes resulting from nonlinearity
- high degrees of localization in space and/or time

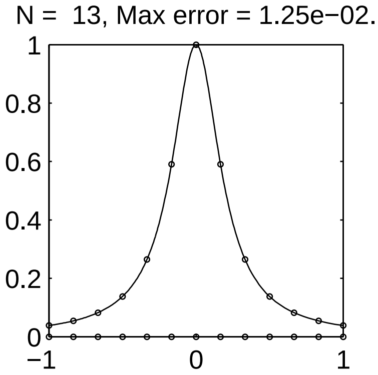
Residual Subsampling Scheme



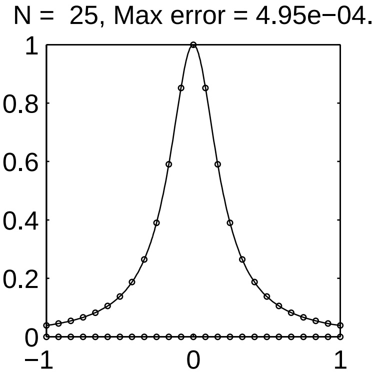
Goal

Obtain an accurate solution using a minimal number of automatically chosen nodes.

Runge Function

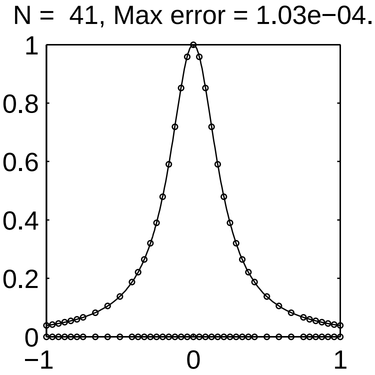



```
%----------------MATLAB CODE----------------
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r,epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(-1,1,N)';
ref = true;
while any(ref)
  N = length(x); dx = diff(x);
  epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
  y = x(1:N-1) + 0.5*dx;
  A = zeros(N); B = zeros(N-1,N);
  for j=1:N
    A(:,j) = phi(x-x(j),epsilon(j));
    B(:,j) = phi(y-x(j),epsilon(j));
  end
  lambda = A\f(x); resid = abs(B*lambda-f(y));
  ref = resid > thetar; x = sort([x;y(ref)]);
  coarsen = resid(1:N-2) < thetac & ...
            resid(2:N-1) < thetac;
  coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

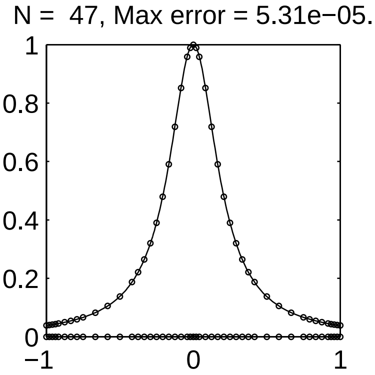



```
%-----------------MATLAB CODE-----------------
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r,epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(-1,1,N)';
ref = true;
while any(ref)
  N = length(x); dx = diff(x);
  epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
  y = x(1:N-1) + 0.5*dx;
  A = zeros(N); B = zeros(N-1,N);
  for j=1:N
    A(:,j) = phi(x-x(j),epsilon(j));
    B(:,j) = phi(y-x(j),epsilon(j));
  end
  lambda = A\f(x); resid = abs(B*lambda-f(y));
  ref = resid > thetar; x = sort([x;y(ref)]);
  coarsen = resid(1:N-2) < thetac & ...
            resid(2:N-1) < thetac;
  coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

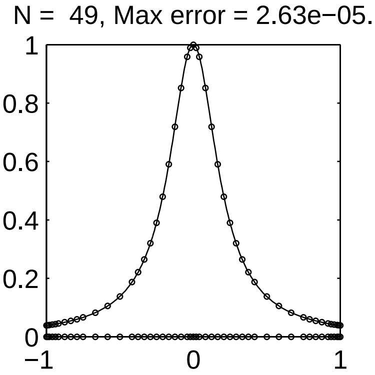
Runge Function




```
%----------------MATLAB CODE----------------
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r,epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(-1,1,N)';
ref = true;
while any(ref)
  N = length(x); dx = diff(x);
  epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
  y = x(1:N-1) + 0.5*dx;
  A = zeros(N); B = zeros(N-1,N);
  for j=1:N
    A(:,j) = phi(x-x(j),epsilon(j));
    B(:,j) = phi(y-x(j),epsilon(j));
  end
  lambda = A\f(x); resid = abs(B*lambda-f(y));
  ref = resid > thetar; x = sort([x;y(ref)]);
  coarsen = resid(1:N-2) < thetac & ...
            resid(2:N-1) < thetac;
  coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

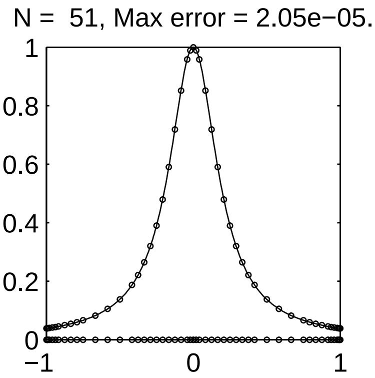



```
%----------------MATLAB CODE----------------
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r,epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(-1,1,N)';
ref = true;
while any(ref)
  N = length(x); dx = diff(x);
  epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
  y = x(1:N-1) + 0.5*dx;
  A = zeros(N); B = zeros(N-1,N);
  for j=1:N
    A(:,j) = phi(x-x(j),epsilon(j));
    B(:,j) = phi(y-x(j),epsilon(j));
  end
  lambda = A\f(x); resid = abs(B*lambda-f(y));
  ref = resid > thetar; x = sort([x;y(ref)]);
  coarsen = resid(1:N-2) < thetac & ...
            resid(2:N-1) < thetac;
  coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

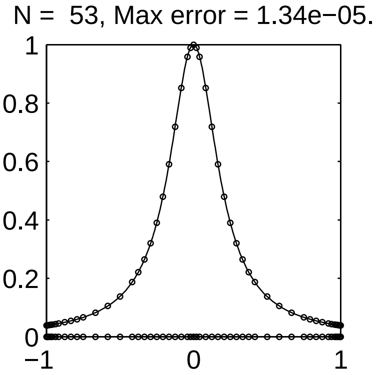



```
%----------------MATLAB CODE----------------
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r,epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(-1,1,N)';
ref = true;
while any(ref)
  N = length(x); dx = diff(x);
  epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
  y = x(1:N-1) + 0.5*dx;
  A = zeros(N); B = zeros(N-1,N);
  for j=1:N
    A(:,j) = phi(x-x(j),epsilon(j));
    B(:,j) = phi(y-x(j),epsilon(j));
  end
  lambda = A\f(x); resid = abs(B*lambda-f(y));
  ref = resid > thetar; x = sort([x;y(ref)]);
  coarsen = resid(1:N-2) < thetac & ...
            resid(2:N-1) < thetac;
  coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

Runge Function

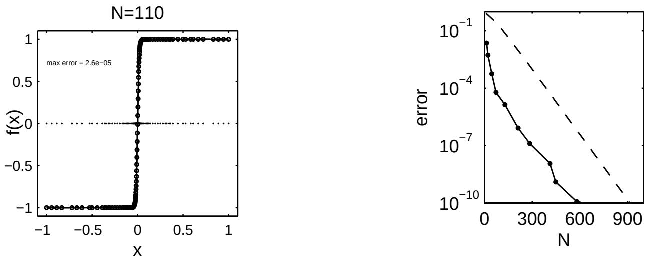



```
%------------------MATLAB CODE------------------
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r,epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(-1,1,N)';
ref = true;
while any(ref)
  N = length(x); dx = diff(x);
  epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
  y = x(1:N-1) + 0.5*dx;
  A = zeros(N); B = zeros(N-1,N);
  for j=1:N
    A(:,j) = phi(x-x(j),epsilon(j));
    B(:,j) = phi(y-x(j),epsilon(j));
  end
  lambda = A\f(x); resid = abs(B*lambda-f(y));
  ref = resid > thetar; x = sort([x;y(ref)]);
  coarsen = resid(1:N-2) < thetac & ...
            resid(2:N-1) < thetac;
  coarsen = 1+find(coarsen); x(coarsen) = [];
end
```




```
%----------------------MATLAB CODE----------------------
thetar = 2e-5; thetac = 1e-8; N = 13;
f = @(x) 1./(1+25*x.^2);
phi = @(r,epsilon) sqrt((epsilon*r).^2 + 1);
x = linspace(-1,1,N)';
ref = true;
while any(ref)
  N = length(x); dx = diff(x);
  epsilon = 0.75*min([Inf;1./dx],[1./dx;Inf]);
  y = x(1:N-1) + 0.5*dx;
  A = zeros(N); B = zeros(N-1,N);
  for j=1:N
    A(:,j) = phi(x-x(j),epsilon(j));
    B(:,j) = phi(y-x(j),epsilon(j));
  end
  lambda = A\f(x); resid = abs(B*lambda-f(y));
  ref = resid > thetar; x = sort([x;y(ref)]);
  coarsen = resid(1:N-2) < thetac & ...
            resid(2:N-1) < thetac;
  coarsen = 1+find(coarsen); x(coarsen) = [];
end
```

tanh(60x − .01)

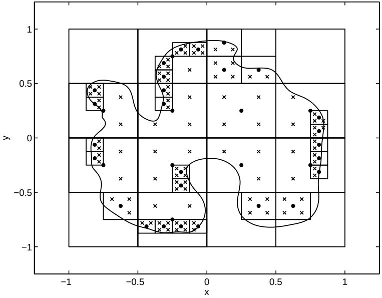


It	N	N_c	N_r	$\kappa(A)$	$\|\|.\|\|_\infty$
1	11	0	18	5.090e+02	7.9211e-01
2	29	0	34	2.632e+04	4.1501e-01
3	63	0	31	4.545e+05	1.2544e-01
4	94	3	30	4.330e+06	1.1129e-02
5	121	4	12	3.962e+07	2.6766e-04
6	129	2	2	3.180e+08	5.7980e-05
7	129	0	0	3.038e+08	2.5329e-05

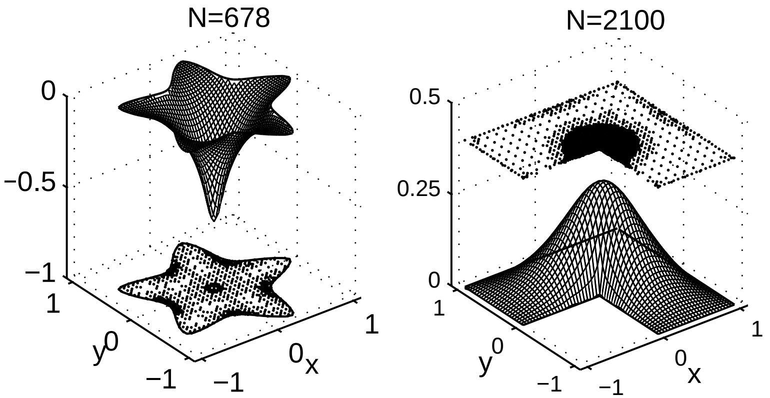
N_r ,N_c = Number of centers to be added/removed respectively.

Scheme

1. Initial coarse collection of nonoverlapping regular boxes in R^d that cover the domain Ω of interest.
2. Geometric adaptation.
3. Refi ning/Coarsening steps



Poisson Equation with Dirichlet condition



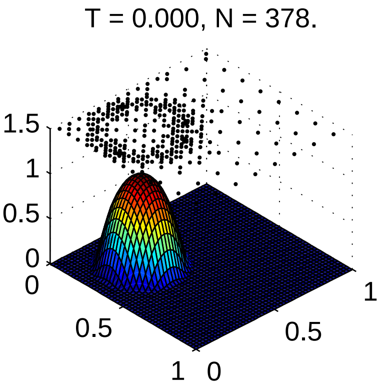
Burgers' Equation

$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$

$$u = 0 \text{ on } \partial\Omega$$

$$f(u) = \tfrac{1}{2}u^2$$

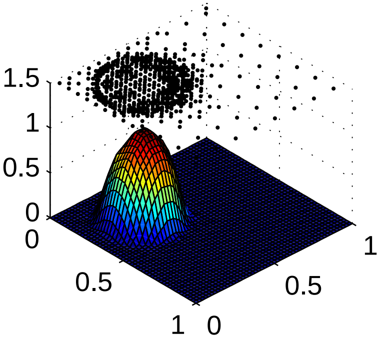
$$\nu = 2.10^{-3}$$



Burgers' Equation

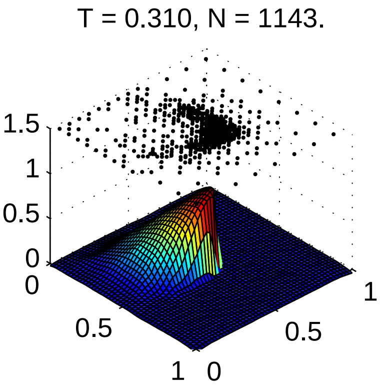
$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$
$$u = 0 \text{ on } \partial\Omega$$
$$f(u) = \tfrac{1}{2}u^2$$
$$\nu = 2.10^{-3}$$

T = 0.010, N = 764.



Burgers' Equation

$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$
$$u = 0 \text{ on } \partial\Omega$$
$$f(u) = \tfrac{1}{2}u^2$$
$$\nu = 2.10^{-3}$$



Burgers' Equation

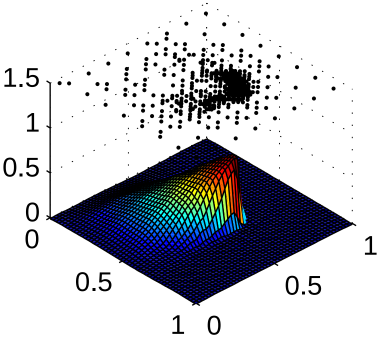
$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$

$$u = 0 \text{ on } \partial\Omega$$

$$f(u) = \frac{1}{2}u^2$$

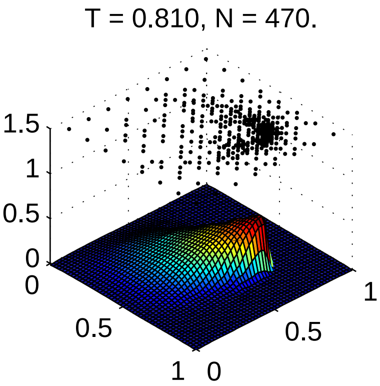
$$\nu = 2.10^{-3}$$

T = 0.510, N = 710.



Burgers' Equation

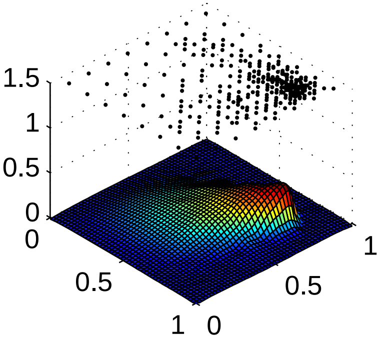
$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$
$$u = 0 \text{ on } \partial\Omega$$
$$f(u) = \tfrac{1}{2}u^2$$
$$\nu = 2.10^{-3}$$



Burgers' Equation

$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$
$$u = 0 \text{ on } \partial\Omega$$
$$f(u) = \tfrac{1}{2}u^2$$
$$\nu = 2.10^{-3}$$

T = 1.190, N = 356.



Buckley-Leverett

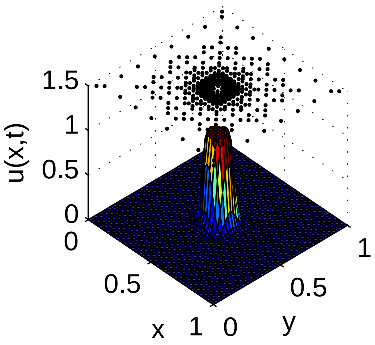
$$\nu\Delta u - \nabla f(u)\cdot\vec{n} = \frac{\partial u}{\partial t}$$

$$u = 0 \text{ on } \partial\Omega$$

$$f(u) = \frac{u^2}{u^2 + \mu(1-u)^2}$$

$$\nu = 10^{-3}\mu = \frac{1}{2}$$

T = 0.000, N = 484.



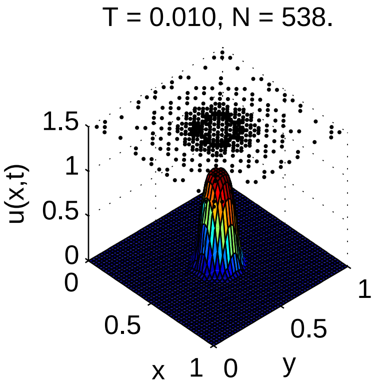
Buckley-Leverett

$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$

$$u = 0 \text{ on } \partial\Omega$$

$$f(u) = \frac{u^2}{u^2 + \mu(1-u)^2}$$

$$\nu = 10^{-3} \mu = \frac{1}{2}$$



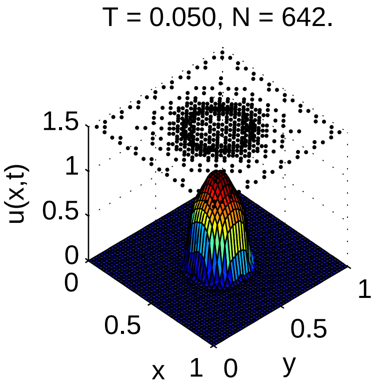
Buckley-Leverett

$$\nu\Delta u - \nabla f(u)\cdot\vec{n} = \frac{\partial u}{\partial t}$$

$$u = 0 \text{ on } \partial\Omega$$

$$f(u) = \frac{u^2}{u^2 + \mu(1-u)^2}$$

$$\nu = 10^{-3}\mu = \frac{1}{2}$$



Step and Adapt / Method of Lines

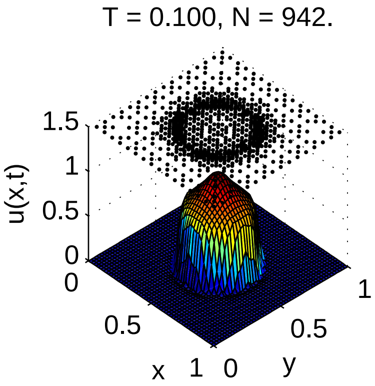
Buckley-Leverett

$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$

$$u = 0 \text{ on } \partial\Omega$$

$$f(u) = \frac{u^2}{u^2 + \mu(1-u)^2}$$

$$\nu = 10^{-3} \mu = \frac{1}{2}$$



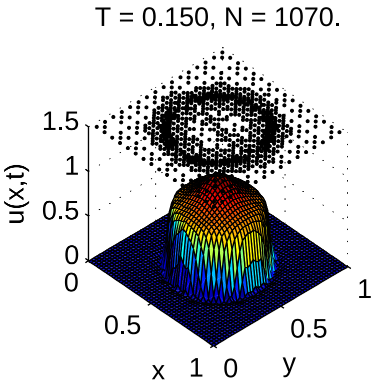
Buckley-Leverett

$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$

$$u = 0 \text{ on } \partial\Omega$$

$$f(u) = \frac{u^2}{u^2 + \mu(1-u)^2}$$

$$\nu = 10^{-3} \mu = \tfrac{1}{2}$$



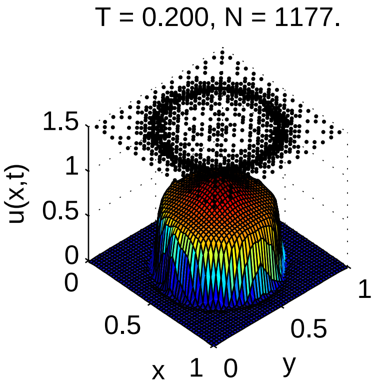
Buckley-Leverett

$$\nu \Delta u - \nabla f(u) \cdot \vec{n} = \frac{\partial u}{\partial t}$$

$$u = 0 \text{ on } \partial\Omega$$

$$f(u) = \frac{u^2}{u^2 + \mu(1-u)^2}$$

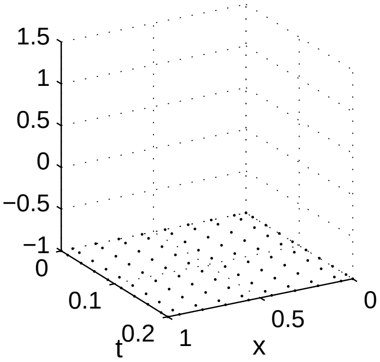
$$\nu = 10^{-3} \mu = \frac{1}{2}$$



1-D Burgers' Equation

$$\nu u_{xx} - u u_x = u_t, \qquad 0 < x < 1$$
$$u(0,t) = u(1,t) = 0$$
$$u(x,0) = \sin(2\pi x) + \tfrac{1}{2}\sin(\pi x).$$

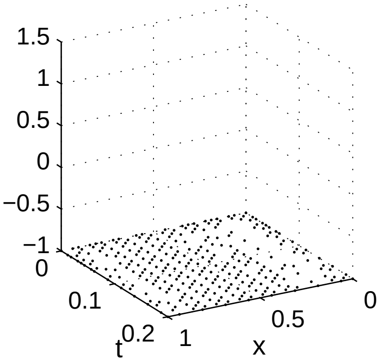
where, $\nu = 10^{-3}$



1-D Burgers' Equation

$$\nu u_{xx} - u u_x = u_t, \qquad 0 < x < 1$$
$$u(0,t) = u(1,t) = 0$$
$$u(x,0) = \sin(2\pi x) + \tfrac{1}{2}\sin(\pi x).$$

where, $\nu = 10^{-3}$



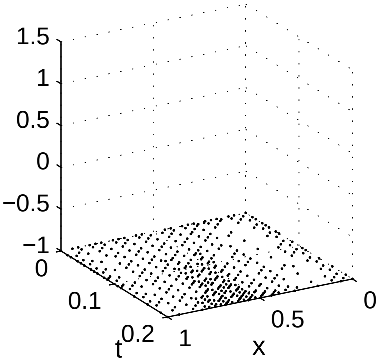
1-D Burgers' Equation

$$\nu u_{xx} - u u_x = u_t, \qquad 0 < x < 1$$

$$u(0,t) = u(1,t) = 0$$

$$u(x,0) = \sin(2\pi x) + \tfrac{1}{2}\sin(\pi x).$$

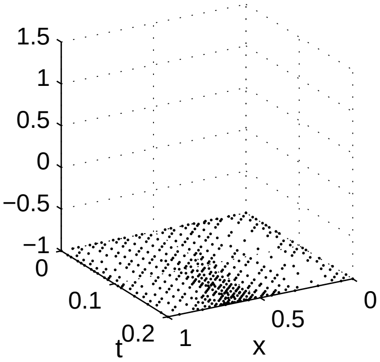
where, $\nu = 10^{-3}$



1-D Burgers' Equation

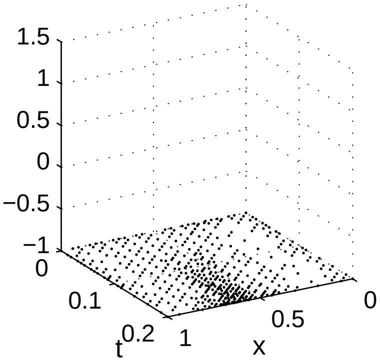
$$\nu u_{xx} - u u_x = u_t, \qquad 0 < x < 1$$
$$u(0,t) = u(1,t) = 0$$
$$u(x,0) = \sin(2\pi x) + \tfrac{1}{2}\sin(\pi x).$$

where, $\nu = 10^{-3}$



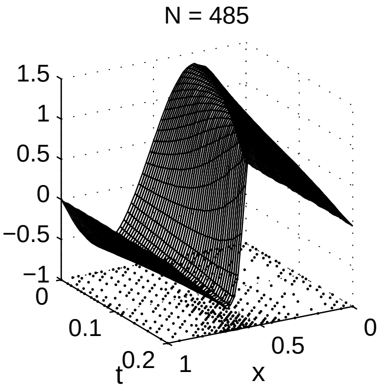
1-D Burgers' Equation

$$\nu u_{xx} - u u_x = u_t, \qquad 0 < x < 1$$
$$u(0,t) = u(1,t) = 0$$
$$u(x,0) = \sin(2\pi x) + \tfrac{1}{2}\sin(\pi x).$$
where, $\nu = 10^{-3}$



1-D Burgers' Equation

$$\nu u_{xx} - u u_x = u_t, \qquad 0 < x < 1$$
$$u(0,t) = u(1,t) = 0$$
$$u(x,0) = \sin(2\pi x) + \tfrac{1}{2}\sin(\pi x).$$
where, $\nu = 10^{-3}$



Things to be done

- Theory and model problems.
- Stability and Accuracy.
- Finding the best way to choose shape parameters.
- Applications (e.g. Lubrication theory in human eye).