AMCS 394E: Contemp. Topics in Computational Science.
Computing with the finite element method

David I. Ketcheson and Manuel Quezada de Luna

King Abdullah University of Science and Technology

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\mathrm{div}, \Omega)$, $H(\mathrm{curl}, \Omega)$, $\ldots$.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\text{div}, \Omega)$, $H(\text{curl}, \Omega)$, ...
* We are using conforming spaces; i.e., $V_h \subset V$.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\text{div}, \Omega)$, $H(\text{curl}, \Omega)$, ...
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\text{div}, \Omega)$, $H(\text{curl}, \Omega)$, $\ldots$.
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\mathrm{div}, \Omega)$, $H(\mathrm{curl}, \Omega)$, ...
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
  - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\mathrm{div}, \Omega)$, $H(\mathrm{curl}, \Omega)$, ...
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
  - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.
* We are using C^0 p.w. polynomial approximations $v|_K \in \mathbb{P}^p$.

Summary of some decisions we have taken

- We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\text{div}, \Omega)$, $H(\text{curl}, \Omega)$, $\ldots$.
- We are using conforming spaces; i.e., $V_h \subset V$.
- We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
  - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.
- We are using C^0 p.w. polynomial approximations $v|_K \in \mathbb{P}^p$.
  - Other options include the use of B-splines, Fourier basis, RT-FEs, Nedelec FEs, etc.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
    - Other popular options: $H(\mathrm{div}, \Omega)$, $H(\mathrm{curl}, \Omega)$, ...
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.
    - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
    - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.
* We are using C^0 p.w. polynomial approximations $v|_K \in \mathbb{P}^p$.
    - Other options include the use of B-splines, Fourier basis, RT-FEs, Nedelec FEs, etc.
* We are considering the standard Galerkin finite elements.

Summary of some decisions we have taken

- We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\text{div}, \Omega)$, $H(\text{curl}, \Omega)$, ...
- We are using conforming spaces; i.e., $V_h \subset V$.
- We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
  - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.
- We are using C^0 p.w. polynomial approximations $v|_K \in \mathbb{P}^p$.
  - Other options include the use of B-splines, Fourier basis, RT-FEs, Nedelec FEs, etc.
- We are considering the standard Galerkin finite elements.
  - Here the test and trial functions are approximated by the same space V_h.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\mathrm{div}, \Omega)$, $H(\mathrm{curl}, \Omega)$, ...
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
  - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.
* We are using C^0 p.w. polynomial approximations $v|_K \in \mathbb{P}^p$.
  - Other options include the use of B-splines, Fourier basis, RT-FEs, Nedelec FEs, etc.
* We are considering the standard Galerkin finite elements.
  - Here the test and trial functions are approximated by the same space V_h.
  - Petrov Galerkin FEs: use different spaces of approximation.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\text{div}, \Omega)$, $H(\text{curl}, \Omega)$, $\ldots$.
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
  - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.
* We are using C^0 p.w. polynomial approximations $v|_K \in \mathbb{P}^p$.
  - Other options include the use of B-splines, Fourier basis, RT-FEs, Nedelec FEs, etc.
* We are considering the standard Galerkin finite elements.
  - Here the test and trial functions are approximated by the same space V_h.
  - Petrov Galerkin FEs: use different spaces of approximation.
* We are using linear spaces to approximate the mesh.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\mathrm{div}, \Omega)$, $H(\mathrm{curl}, \Omega)$, ...
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
  - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.
* We are using C^0 p.w. polynomial approximations $v|_K \in \mathbb{P}^p$.
  - Other options include the use of B-splines, Fourier basis, RT-FEs, Nedelec FEs, etc.
* We are considering the standard Galerkin finite elements.
  - Here the test and trial functions are approximated by the same space V_h.
  - Petrov Galerkin FEs: use different spaces of approximation.
* We are using linear spaces to approximate the mesh.
  - We could use higher-order approximations to include curved meshes.

Summary of some decisions we have taken

* We are using $H^1(\Omega)$ spaces to approx. the weak solution.
  - Other popular options: $H(\text{div}, \Omega)$, $H(\text{curl}, \Omega)$, ...
* We are using conforming spaces; i.e., $V_h \subset V$.
* We are considering $C^0(\Omega)$ FE spaces.
  - Discontinuous Galerkin FEs are popular to solve hyperbolic equations.
  - Higher smoothness is also possible; i.e., $C^1(\Omega)$ (or higher) FEs.
* We are using C^0 p.w. polynomial approximations $v|_K \in \mathbb{P}^p$.
  - Other options include the use of B-splines, Fourier basis, RT-FEs, Nedelec FEs, etc.
* We are considering the standard Galerkin finite elements.
  - Here the test and trial functions are approximated by the same space V_h.
  - Petrov Galerkin FEs: use different spaces of approximation.
* We are using linear spaces to approximate the mesh.
  - We could use higher-order approximations to include curved meshes.
* We are assuming Ω and the basis functions don't change in time.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

* Consider the (cont. and discrete) 'weak' formulations.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

* Consider the (cont. and discrete) ‘weak’ formulations.
* Obtain the linear algebra problem associated with the projection.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

* Consider the (cont. and discrete) ‘weak’ formulations.
* Obtain the linear algebra problem associated with the projection.
* See the 1D and the 2D codes in Matlab.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

* Consider the (cont. and discrete) ‘weak’ formulations.
* Obtain the linear algebra problem associated with the projection.
* See the 1D and the 2D codes in Matlab.
* See how to compute the errors and convergence tables.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

- Consider the (cont. and discrete) ‘weak’ formulations.
- Obtain the linear algebra problem associated with the projection.
- See the 1D and the 2D codes in Matlab.
- See how to compute the errors and convergence tables.
- A note about plotting.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

* Consider the (cont. and discrete) ‘weak’ formulations.
* Obtain the linear algebra problem associated with the projection.
* See the 1D and the 2D codes in Matlab.
* See how to compute the errors and convergence tables.
* A note about plotting.

Solve the Poisson equation in 2D.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

* Consider the (cont. and discrete) ‘weak’ formulations.
* Obtain the linear algebra problem associated with the projection.
* See the 1D and the 2D codes in Matlab.
* See how to compute the errors and convergence tables.
* A note about plotting.

Solve the Poisson equation in 2D.

* Consider the (cont. and discrete) ‘weak’ formulations.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

* Consider the (cont. and discrete) 'weak' formulations.
* Obtain the linear algebra problem associated with the projection.
* See the 1D and the 2D codes in Matlab.
* See how to compute the errors and convergence tables.
* A note about plotting.

Solve the Poisson equation in 2D.

* Consider the (cont. and discrete) 'weak' formulations.
* Obtain the linear algebra problem associated with the problem.

Implementing the FE method in Matlab

In this class we will see some 1D and 2D implementations in Matlab.

Project some functions u into the space V_h.

* Consider the (cont. and discrete) ‘weak’ formulations.
* Obtain the linear algebra problem associated with the projection.
* See the 1D and the 2D codes in Matlab.
* See how to compute the errors and convergence tables.
* A note about plotting.

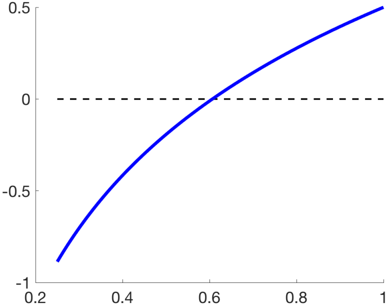
Solve the Poisson equation in 2D.

* Consider the (cont. and discrete) ‘weak’ formulations.
* Obtain the linear algebra problem associated with the problem.
* See the code in Matlab.

Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$

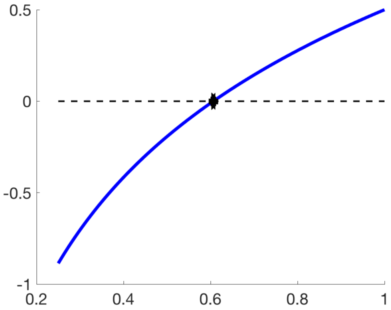


Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$

* We want to find u_* such that $r(u_*) = 0$.

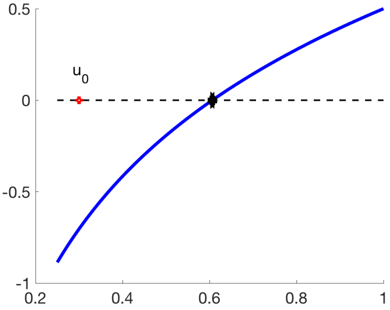


Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$

* Consider an initial guess u_0.

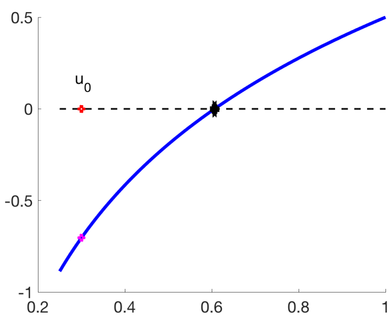


Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$

* Perform a linear approximation of $r(u)$ around u_0.

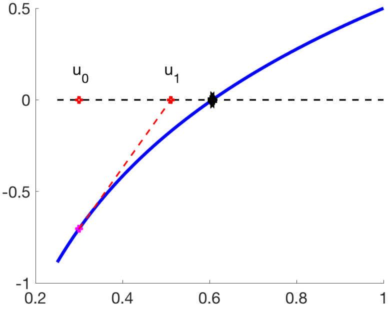


Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$

* Find the root of the linear approximation and call that u_1.

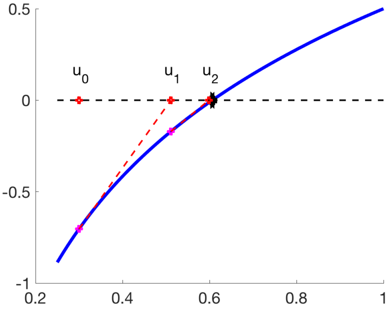


Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$

* Repeat the process until 'convergence is achieved'.



Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$
The $k+1$ iteration is given by

$$u_{k+1} = u_k - (r'(u_k))^{-1} r(u_k).$$

Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$

The $k+1$ iteration is given by

$$u_{k+1} = u_k - (r'(u_k))^{-1} r(u_k).$$

Newton's method for the system of algebraic equations $R(U)$

The $k+1$ iteration is given by

$$U_{k+1} = U_k - \mathrm{inv}\Big(\frac{\partial R(U_k)}{\partial U_k}\Big) R(U_k).$$

Nonlinear elliptic PDE via Newton's method

In this class we will see how to use Newton's method to solve a nonlinear elliptic equation.

Newton's method for the algebraic scalar equation $r(u)$

The $k+1$ iteration is given by

$$u_{k+1} = u_k - (r'(u_k))^{-1} r(u_k).$$

Newton's method for the system of algebraic equations $R(U)$

The $k+1$ iteration is given by

$$U_{k+1} = U_k - \mathrm{inv}\Bigg(\underbrace{\frac{\partial R(U_k)}{\partial U_k}}_{\text{Jacobian}}\Bigg) R(U_k).$$

Nonlinear elliptic PDE via Newton's method

Consider the following (discrete) weak formulation.

$$\alpha m_i^{\Gamma(\tilde{u}_h)}(U_i - \tilde{U}_i) + \int_\Omega S_\epsilon(\tilde{u}_h)\left[|\nabla u_h| - 1\right]\phi d\mathbf{x} + c\int_\Omega h(\mathbf{x})\nabla u_h \cdot \nabla\phi d\mathbf{x} = 0$$

Nonlinear elliptic PDE via Newton's method

Consider the following (discrete) weak formulation.

$$\alpha m_i^{\Gamma(\tilde{u}_h)}(U_i - \tilde{U}_i) + \int_\Omega S_\epsilon(\tilde{u}_h)\left[|\nabla u_h| - 1\right]\phi d\mathbf{x} + c\underbrace{\int_\Omega h(\mathbf{x})\nabla u_h \cdot \nabla\phi d\mathbf{x}}_{\text{stabilization}} = 0$$

Nonlinear elliptic PDE via Newton's method

Consider the following (discrete) weak formulation.

$$\underbrace{\alpha m_i^{\Gamma(\tilde{u}_h)}(U_i - \tilde{U}_i)}_{\text{penalization}} + \int_\Omega S_\epsilon(\tilde{u}_h)\left[|\nabla u_h| - 1\right]\phi d\mathbf{x} + c\int_\Omega h(\mathbf{x})\nabla u_h \cdot \nabla\phi d\mathbf{x} = 0$$

Nonlinear elliptic PDE via Newton's method

Consider the following (discrete) weak formulation.

$$\alpha m_i^{\Gamma(\tilde{u}_h)}(U_i - \tilde{U}_i) + \int_\Omega S_\epsilon(\tilde{u}_h)\left[|\nabla u_h| - 1\right]\phi d\mathbf{x} + c\int_\Omega h(\mathbf{x})\nabla u_h \cdot \nabla\phi d\mathbf{x} = 0$$

Here $\alpha = 1 \times 10^{10}$, and

$$\begin{aligned}\int_{\Gamma(\tilde{u})}(u_h - \tilde{u}_h)\phi_i d\mathbf{x} &\approx \int_\Omega \delta_\epsilon(\tilde{u})(u_h - \tilde{u}_h)\phi_i d\mathbf{x} \\ &= \sum_j (U_j - \tilde{U}_j)\int_\Omega \delta_\epsilon(\tilde{u})\phi_i\phi_j d\mathbf{x} \\ &\approx (U_i - \tilde{U}_i)\int_\Omega \delta_\epsilon(\tilde{u})\phi_i d\mathbf{x} = m_i^{\Gamma(\tilde{u})}(U_i - \tilde{U}_i)\end{aligned}$$

is a penalization term that forces the solution u_h to be close to $\tilde{u}_h$

Nonlinear elliptic PDE via Newton's method

Consider the following (discrete) weak formulation.

$$\alpha m_i^{\Gamma(\tilde{u}_h)}(U_i - \tilde{U}_i) + \underbrace{\int_\Omega S_\epsilon(\tilde{u}_h)\left[|\nabla u_h| - 1\right]\phi d\mathbf{x}}_{\text{weak signed Eikonal equation}} + c\int_\Omega h(\mathbf{x})\nabla u_h \cdot \nabla\phi d\mathbf{x} = 0$$

Nonlinear elliptic PDE via Newton's method

The residual (within Newton's method) is

$$R(U) = \alpha m_i^{\Gamma(\tilde{u}_h)}(U_i - \tilde{U}_i) + \underbrace{\int_\Omega S_\epsilon(\tilde{u}_h)\left[|\nabla u_h| - 1\right]\phi d\mathbf{x}}_{=:E(U)} + S^h U$$

Nonlinear elliptic PDE via Newton's method

The residual (within Newton's method) is

$$R(U) = \alpha m_i^{\Gamma(\tilde{u}_h)}(U_i - \tilde{U}_i) + E(U) + S^h U$$

Nonlinear elliptic PDE via Newton's method

The Jacobian (within Newton's method) is given by

$$J = \frac{\partial R(U)}{\partial U} = \frac{\partial}{\partial U}\left[\alpha M_L^{\Gamma(\tilde{u}_h)}(U - \tilde{U}) + E(U) + S^h U\right]$$

Nonlinear elliptic PDE via Newton's method

The Jacobian (within Newton's method) is given by

$$J = \frac{\partial R(U)}{\partial U} = \frac{\partial}{\partial U}\left[\alpha M_L^{\Gamma(\tilde{u}_h)}(U - \tilde{U}) + E(U) + S^h U\right]$$

$$= \alpha M_L^{\Gamma(\tilde{u}_h)} + \frac{\partial E(U)}{\partial U} + S^h$$

Nonlinear elliptic PDE via Newton's method

The Jacobian (within Newton's method) is given by

$$J = \frac{\partial R(U)}{\partial U} = \frac{\partial}{\partial U}\left[\alpha M_L^{\Gamma(\tilde{u}_h)}(U - \tilde{U}) + E(U) + S^h U\right]$$

$$= \alpha M_L^{\Gamma(\tilde{u}_h)} + \frac{\partial E(U)}{\partial U} + S^h$$

Let us work out the details to get

$$\frac{\partial E(U)}{\partial U} = \dots$$

Nonlinear elliptic PDE via Newton's method

Once we have the residual and Jacobian for a given iteration U^k, we proceed as follows:

Nonlinear elliptic PDE via Newton's method

Once we have the residual and Jacobian for a given iteration U^k, we proceed as follows:

Consider an initial guess U^0

Nonlinear elliptic PDE via Newton's method

Once we have the residual and Jacobian for a given iteration U^k, we proceed as follows:

Consider an initial guess U^0
Set $U^k = U^0$

Nonlinear elliptic PDE via Newton's method

Once we have the residual and Jacobian for a given iteration U^k, we proceed as follows:

Consider an initial guess U^0
Set $U^k = U^0$
Compute $R^k = R(U^k)$ and norm_res = norm(R^k)

Nonlinear elliptic PDE via Newton's method

Once we have the residual and Jacobian for a given iteration U^k, we proceed as follows:

Consider an initial guess U^0
Set $U^k = U^0$
Compute $R^k = R(U^k)$ and norm_res = norm(R^k)
while norm_res>tol **do**

end while

Nonlinear elliptic PDE via Newton's method

Once we have the residual and Jacobian for a given iteration U^k, we proceed as follows:

Consider an initial guess U^0
Set $U^k = U^0$
Compute $R^k = R(U^k)$ and norm_res = norm(R^k)
while norm_res>tol **do**
    Compute $J^k = J(U^k)$

end while

Nonlinear elliptic PDE via Newton's method

Once we have the residual and Jacobian for a given iteration U^k, we proceed as follows:

Consider an initial guess U^0
Set $U^k = U^0$
Compute $R^k = R(U^k)$ and norm_res = norm(R^k)
while norm_res>tol **do**
  Compute $J^k = J(U^k)$
  Update the solution $U^{k+1} = U^k - \text{inv}(J^k)R(U^k)$

end while

Nonlinear elliptic PDE via Newton's method

Once we have the residual and Jacobian for a given iteration U^k, we proceed as follows:

Consider an initial guess U^0
Set $U^k = U^0$
Compute $R^k = R(U^k)$ and norm_res = norm(R^k)
while norm_res>tol **do**
  Compute $J^k = J(U^k)$
  Update the solution $U^{k+1} = U^k - \text{inv}(J^k)R(U^k)$
  Compute R^{k+1} and norm_res = norm(R^{k+1})
end while