Code verification using the method of manufactured solutions

Jeremy Kozdon

Naval Postgraduate School

2019 January 09

Verification and Validation

Verification

Are we solving equations right?

Validation

Are we solving right equations?

(some) possible verification methods

analytic solutions

measure convergence / error with known solution

code comparison

given two (or more) codes / methods show that they produce similar solutions

self convergence

use a single code at multiple resolutions and show convergence to fixed solution

method of manufactured solutions

assume a solution and add forcing so that this is a solution to a modified equation

ANALYTIC SOLUTIONS

Pros

- error is directly measured
- convergence rate can be estimated

Cons

- typically only simple solutions are available
- deriving solutions can be hard
- may not exercise all aspects of model / code
- forcing boundary data can hide instabilities

CODE COMPARISON

BP1 through BP4 fall into this category

Pros

- uses actual equations / codes
- captures/compares quantities we care about
- cuts through promises of methods to reveal what happens in practice
- uses real(ish) parameters

Cons

- need at least two codes (ideally that solve the exact same equations)
- can be difficult to come up with robust metrics for comparison
- codes often use different computational grids
- just because two codes match, does not mean equations are being solved

SELF CONVERGENCE

If $q(x,t)$ is the true solution and $q_h(x,t)$ the numerical solution, assume the error scales as

$$\varepsilon_h(t) = \|q(\cdot,t) - q_h(\cdot,t)\| \approx C(t)h^p$$

- $C(t)$ is independent of h
- h is some representative resolution measure (grid spacing, element size, etc.)
- $\|\cdot\|$ is some norm of the solution (for instance: L^2, L^∞, normalized l^2, etc.)

If $q(x,t)$ is the true solution and $q_h(x,t)$ the numerical solution, assume the error scales as

$$\varepsilon_h(t) = \|q(\cdot,t) - q_h(\cdot,t)\| \approx C(t)h^p$$

if we know q then with two resolutions h_1 and h_2 we can estimate p

$$p \approx \frac{\log \varepsilon_{h_1} - \log \varepsilon_{h_2}}{\log h_1 - \log h_2}$$

But we do not want to assume we know q...

SELF CONVERGENCE

Two approaches when $q(x,t)$ is not known

- use very small h as proxy for q, e.g, $q(x,t) \approx q_{h_s}(x,t)$
- use three levels: h_1, h_2, and h_3 with $h_1/h_2 = h_2/h_3$

$$p \approx \frac{\log \Delta_{h_1 h_2} - \log \Delta_{h_2 h_3}}{\log h_1 - \log h_2}$$

where $\Delta_{h_1 h_2} = \|q_{h_1}(\cdot,t) - q_{h_2}(\cdot,t)\| \approx C(t) h_1^p (1 - (h_2/h_1)^p)$

SELF CONVERGENCE

Pros

- uses the actual equations / codes
- captures/compares quantities we care about
- uses real(ish) parameters
- can measure(ish) how fast code goes to fixed solution
- only one code is needed

Cons

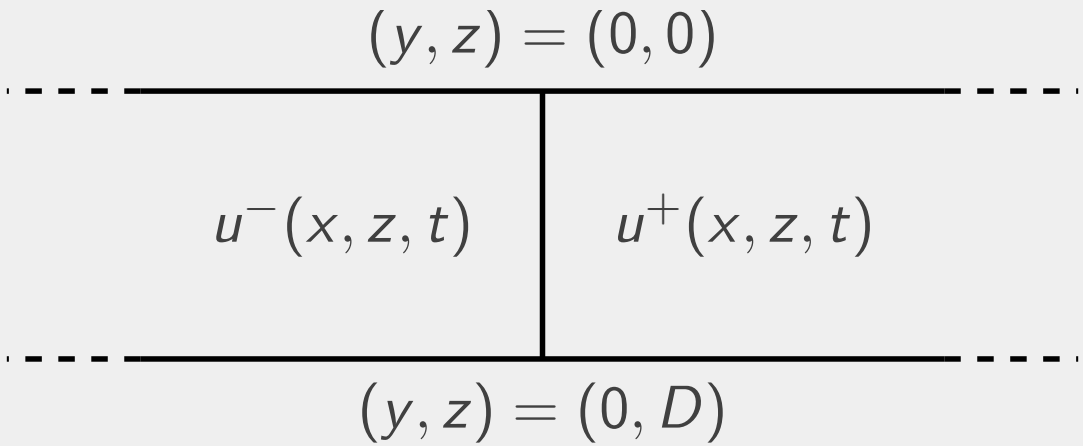
- only shows that code is stable

  Example: won't reveal use of wrong BC or friction law bug
- convergence seen only if all scales resolved

  Example: rupture still discontinuity on mesh even after h^* resolved

Method of manufactured solutions

Assume a solution and then add necessary forcing



$$\Delta u = 0,$$

$$\hat{n} \cdot \nabla u = 0 \qquad \text{on } z = 0, D,$$

$$\tau = \hat{n} \cdot \nabla u^+ = \hat{n} \cdot \nabla u^- \qquad \text{on } x = 0,$$

$$\tau - \eta V = F(V, \psi),$$

$$\frac{\partial \psi}{\partial t} = G(V, \psi)$$

Method of manufactured solutions

Assume solution of the form (more complex solutions are possible![1]):

$$u^{\pm}(x, z, t) = \pm g(t) e^{\mp x} \cos(z)$$

which (I believe) satisfies Laplacian, free surface BC, and continuity of traction...but it will not satisfy the friction law...

[1] Erickson and Dunham (2014) have more complex solution that has multiple length and time scales reminiscent of what comes out of the model, but more complex forcing is required

Method of manufactured solutions

Assume solution of the form (more complex solutions are possible![1]):

$$u^{\pm}(x, z, t) = \pm g(t) e^{\mp x} \cos(z)$$

On the interface:

$$\tau(z, t) = g(t) \cos(z)$$
$$V(z, t) = 2g'(t) \cos(z)$$

$\psi(z, t)$ can then be determined from $\tau - \eta V = F(V, \psi)$.

[1] Erickson and Dunham (2014) have more complex solution that has multiple length and time scales reminiscent of what comes out of the model, but more complex forcing is required

Method of manufactured solutions

Assume solution of the form (more complex solutions are possible![1]):

$$u^{\pm}(x, z, t) = \pm g(t) e^{\mp x} \cos(z)$$

State evolution then modified as follows

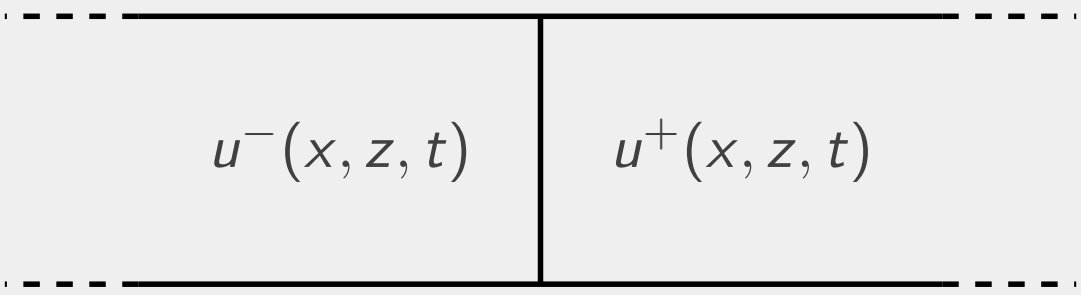
$$\frac{\partial \psi}{\partial t} = G(V, \psi) - G(V^*, \psi^*) + \frac{\partial \psi^*}{\partial t} = G(V, \psi) + s(z, t)$$

superscipt * denotes known functional forms

Importantly state evolution source $s(z, t) = -G(V^*, \psi^*) + \frac{\partial \psi^*}{\partial t}$ is a *known* function of z and t

[1] Erickson and Dunham (2014) have more complex solution that has multiple length and time scales reminiscent of what comes out of the model, but more complex forcing is required

Method of manufactured solutions



modified problem is

$$\Delta u = 0,$$
$$\hat{n} \cdot \nabla u = 0 \qquad \text{on } z = 0, D,$$
$$\tau = \hat{n} \cdot \nabla u^+ = \hat{n} \cdot \nabla u^- \qquad \text{on } x = 0,$$
$$\tau - \eta V = F(V, \psi),$$
$$\frac{\partial \psi}{\partial t} = G(V, \psi) + s(z, t)$$

which has the analytic solution

$$u^{\pm}(x, z, t) = \pm g(t) e^{\mp x} \cos(z)$$

Method of manufactured solutions

Pros

- Solution can be anything
- measures convergence rate and error (code correctness)
- only requires one code
- nice for code testing

Cons

- addition forcing (code modification!)
- can hide numerical instabilities
- scales not always respected
- not a real problem / solution
- good solutions can be hard to find (e.g., respect desired properties)

Other verification techniques

- semianalytic solution comparisons
- preservation of known symmetries
- conservation properties
- ...

Discussion Points

- No one way to do verification, multitude of methods needed!
- Different folks will be convinced by different methods
- Do we want to add some more numerical analysissy style verification problems? (either simple or complex?)
- If we want to do MMS, what restrictions do codes have? (Body forces, domains, boundary conditions)?
- How far could / would we push a self convergence study? (Maybe super upscaled parameters?)
- Other Verification metrics? (Are there must haves for future validation efforts?)
- Might be worth looking outside our community to climate, DOE, and other communities to see how they verify codes
- Thoughts????
- Questions? Talk to Brittany