Title: Computational Relativistic Astrophysics

Date: Feb 09, 2012 02:50 PM

URL: http://pirsa.org/12020148

Abstract:

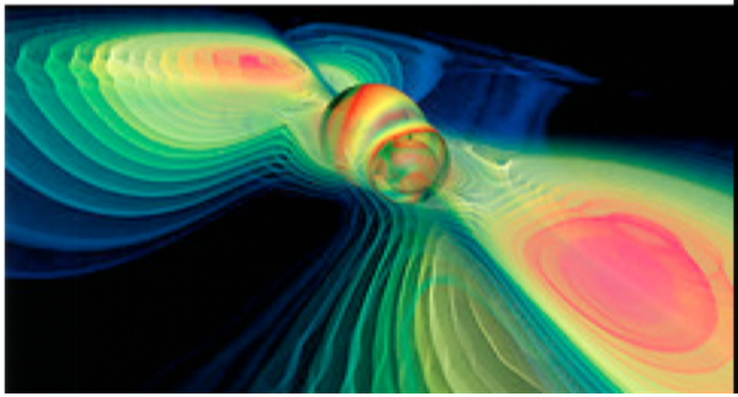
Computational Methods at Perimeter

Perimeter Institute, Waterloo
February 9, 2012

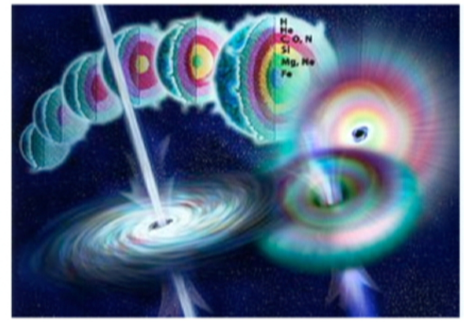
L. F. Richardson
(First to use computers for weather prediction in 1922)

Computational Relativistic Astrophysics

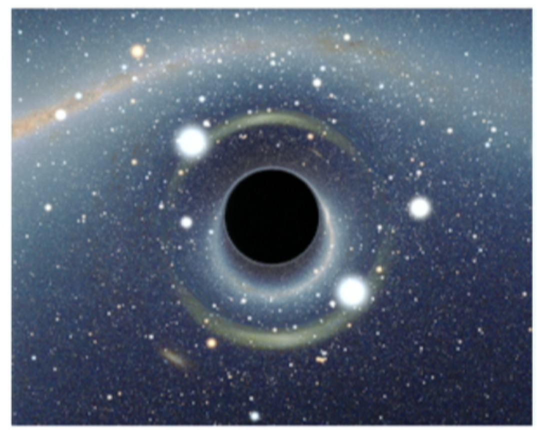
Erik Schnetter, Perimeter Institute
February 9, 2012



Astrophysics Background



- Many astrophysical phenomena ultimately powered by gravity as central engine
- Gravitational waves about to be detected by observatories such as LIGO, GEO600
- Studying compact objects: black holes, neutron stars, also binaries of these

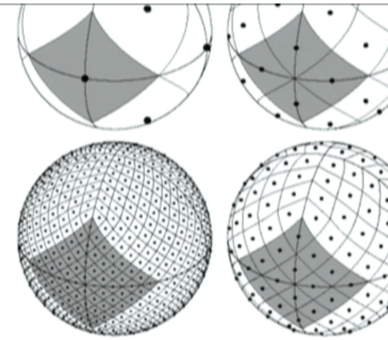



CAUTION

Time Evolution Problems

- Considering here systems of hyperbolic PDEs (i.e. IVPs)
  - (ignoring Lagrangian/Hamiltonian, if any)
  - constraints only monitored during evolution, not directly enforced
  - gauge conditions part of PDE system, not independent (need to have chosen gauge before discussing PDEs)

Discretisation



- Idea: represent (approximate) functions by finite number of degrees of freedom, e.g.:
  - Finite differences: sampling
  - Finite volumes: averaging over cells
  - Finite elements: set of basis functions
  - Spectral methods: spectral coefficients
- Derivative operators become (sparse?) matrices

From Einstein Equations to BSSN

- $G_{ab} = 8\pi T_{ab}$: not directly suitable for numerically solving for metric g_{ab}
  1. Choose time coordinate (aka 3+1 decomposition)
  2. Optionally: convert to first order form (remove second derivatives)
  3. Choose gauge conditions
  4. Modify equations (add new variables, add multiples of constraints, etc.)

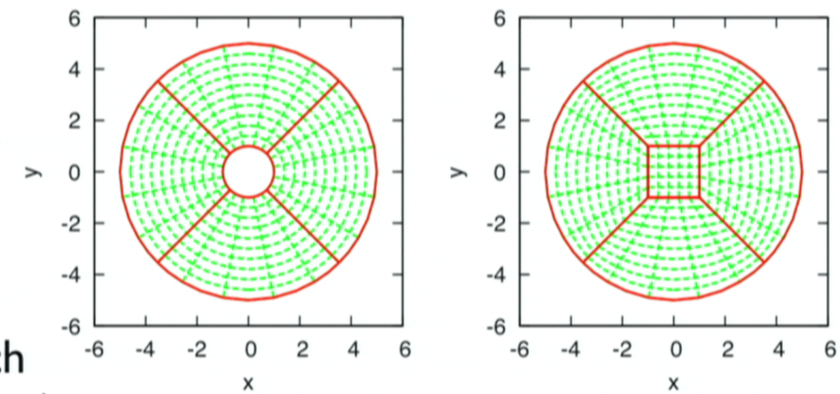
Kranc

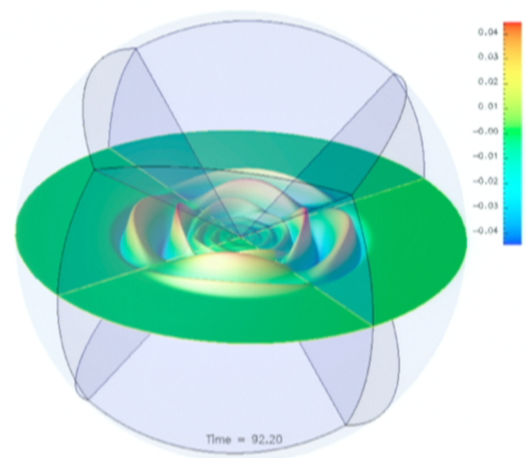
Kranc Assembles Numerical Code

- Einstein equations (BSSN formulation) contain about 5,000 terms, very tedious to code manually
- Want to experiment with different formulations
- Compilers often do not optimise well; need to optimise explicitly (e.g. loop fission, cache tiling)
- Some optimisations are hardware dependent (e.g. vectorisation, accelerators); don't want to repeat this manually for every new system
- See http://kranccode.org/

Multi-Patch Systems

- Basic idea: Cover domain with multiple blocks
- Each block is 3D Cartesian (see 6-patch and 7-patch prototypes)
- Need inter-block boundary conditions

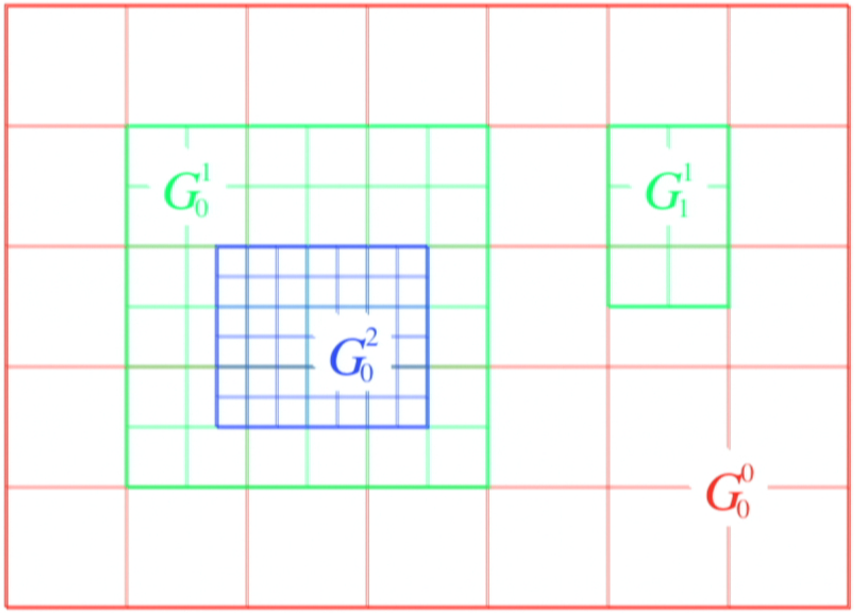




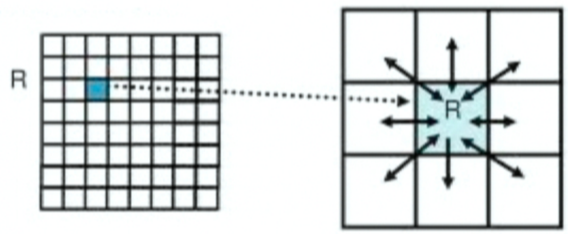
Adaptive Mesh Refinement

Coarse, medium, and fine grids

Grids are aligned

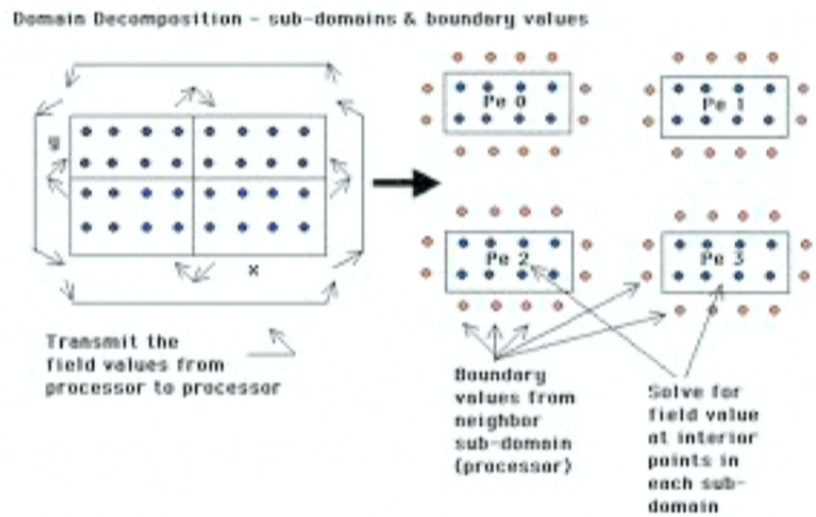


Parallel Processing



2D: Each process communicates data to its 8-neighbors

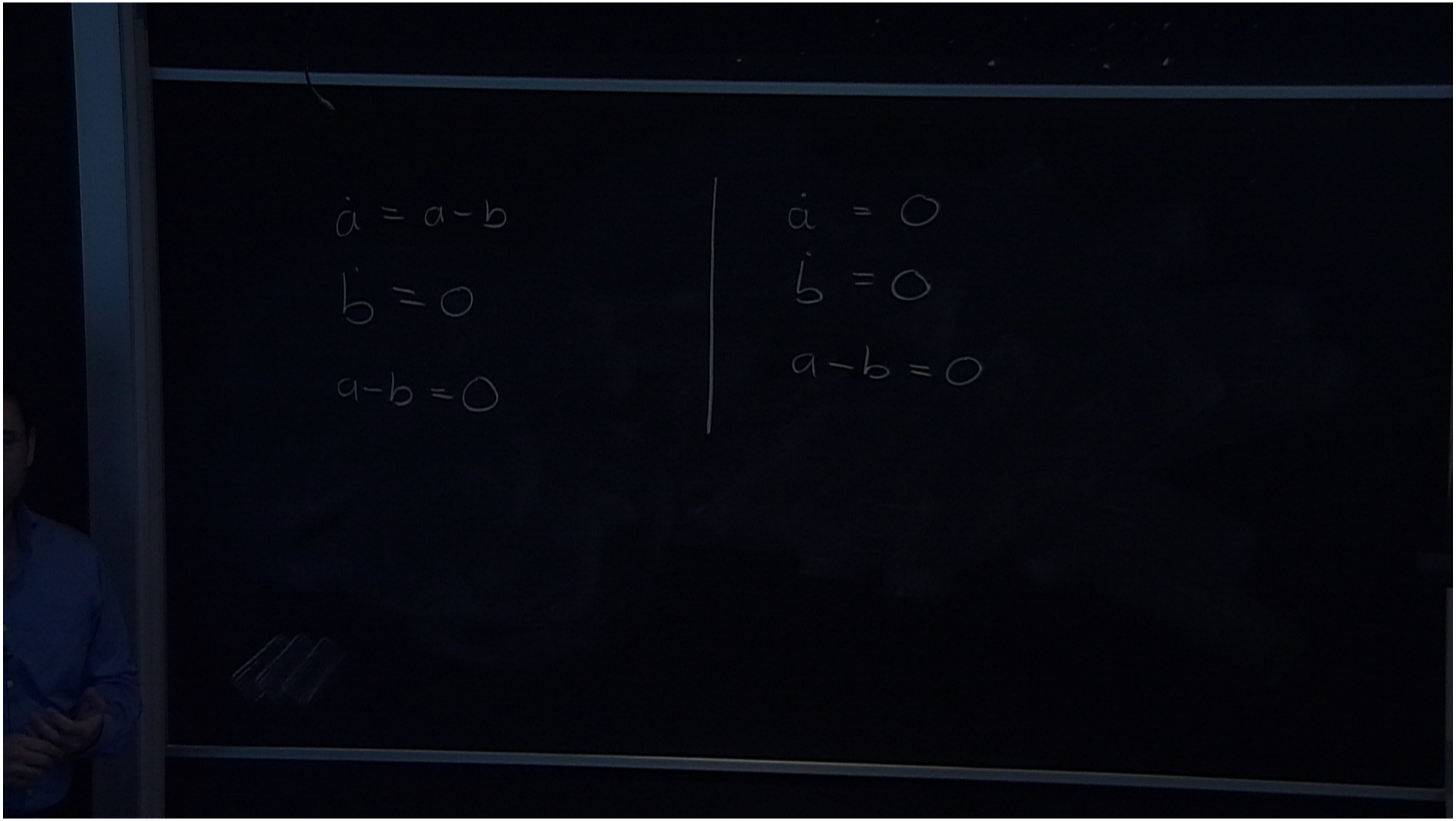
Ghost zones contain copies of neighbouring processes' grid points



Ghost zones are filled via inter-process communication from the corresponding *owner*

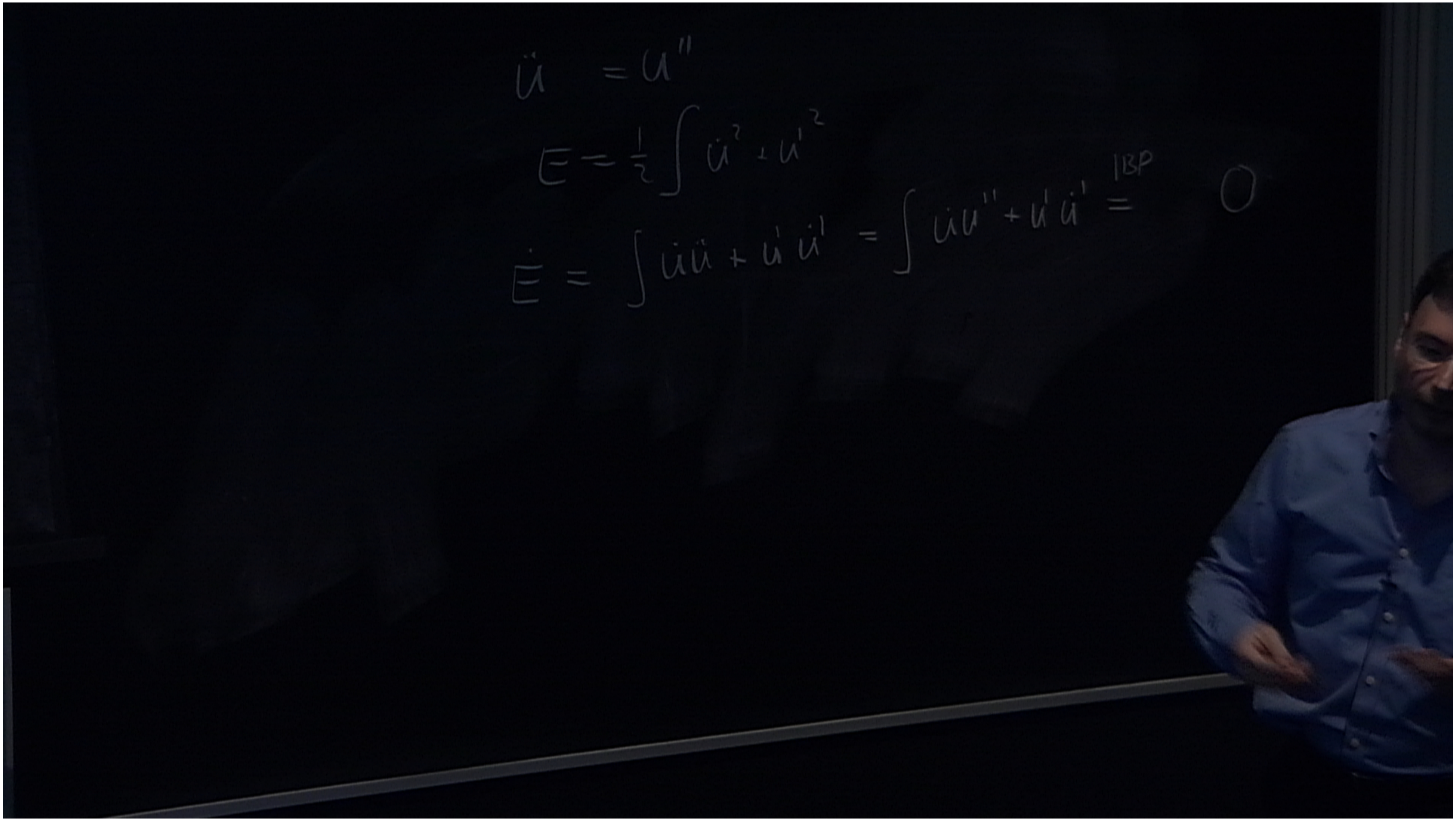
Well-Posedness / Stability

- Difficult to determine whether IVP is well-posed
  - Opinions differ regarding importance of proving well-posedness before attempting a numerical solution
  - One definition for WP: for all times t, norm of solution is bounded by A $\exp(\alpha t)$, with A, α independent of resolution
- Continuum well-posedness $\neq$ discretized well-posedness


$\dot{a} = a - b$
$\dot{b} = 0$
$a - b = 0$
$\dot{a} = 0$
$\dot{b} = 0$
$a - b = 0$

Discrete Stability

- One definition of stability (of a hyperbolic PDE): can define non-negative energy that is non-increasing under time evolution
  - Stability proof typically requires integration by parts
  - Discrete equivalent ("summation by parts") can lead to stable discretization


$\ddot{u} = u''$
$E = \frac{1}{2}\int \dot{u}^2 + u'^2$
$\dot{E} = \int \dot{u}\ddot{u} + u'\dot{u}' = \int \dot{u}u'' + u'\dot{u}' \overset{IBP}{=} 0$