Numerical Methods

Numerisches Praktikum (UKNum) 2023/24

Hubert Klahr Max-Planck-Institut für Astronomie
klahr@mpia.de

Supported by Dhruv Muley,
Michael Cecil and Dominik Ostertag
Lösungen: @mpia.de

http://www.mpia-hd.mpg.de/~klahr/UKNUM2024.html

Numerical Methods

Numerisches Praktikum (UKNum) 2023/24

Three parts today:

1. General Information about the lecture
2. Introduction to Numerical Methods
3. Floating point representation

http://www.mpia-hd.mpg.de/~klahr/UKNUM2024.html

Numerisches Praktikum

Tentative schedule

- Feb. 19th: Lecture 1: Introduction, number representation in a computer
- Feb. 20th: Lecture 2: Interpolation, Extrapolation, Splines
- Feb. 21st: Lecture 3: Solving ordinary differential equations
- Feb. 22nd: Lecture 4: Numerical integration
- Feb. 23rd: Lecture 5: Finding roots, iterative Newton-Raphson method

- Feb. 26th.: Lecture 6: Sort algorithms
- Feb. 27th: Lecture 7: Systems of linear equations
- Feb. 28th: Lecture 8: Statistical methods, data modelling
- Feb. 29th: Lecture 9: Random numbers, Monte Carlo methods
- Mar. 1st: Lecture 10: Summary and concluding remarks

Numerisches Praktikum

Daily schedule:

- 9:15 - 9:45: Presentation of Exercises from Previous Lesson (Students)
- 9:45 - 11:15: Introduction to new Lesson (Lecturers)
- 11:15 - 11:30: Break
- 11:30 - 13:00: Tutorial (Students work with assistance of Lecturers and Tutor)
- Afternoon: Independent Working Time for Students

Solution per Email to the corresponding tutor:
@mpia.de

You can work on the exercises during the tutorial time in presence of lecturers. In the afternoon
you can continue and complete the assigned exercise work.
You can reach us via Email. The preferred channel for questions would be SLACK.

Please write down the results, document
the important bits of the code in proper form (tables, lists, etc). Do not print out the
entire code. The results can be handed in until the following morning 9:15 a.m. by e-mail to
@mpia.de

Criterion for Certificate
60 % of maximum number of points, and
presentation of results at least once.

Maximally 2 students per group!

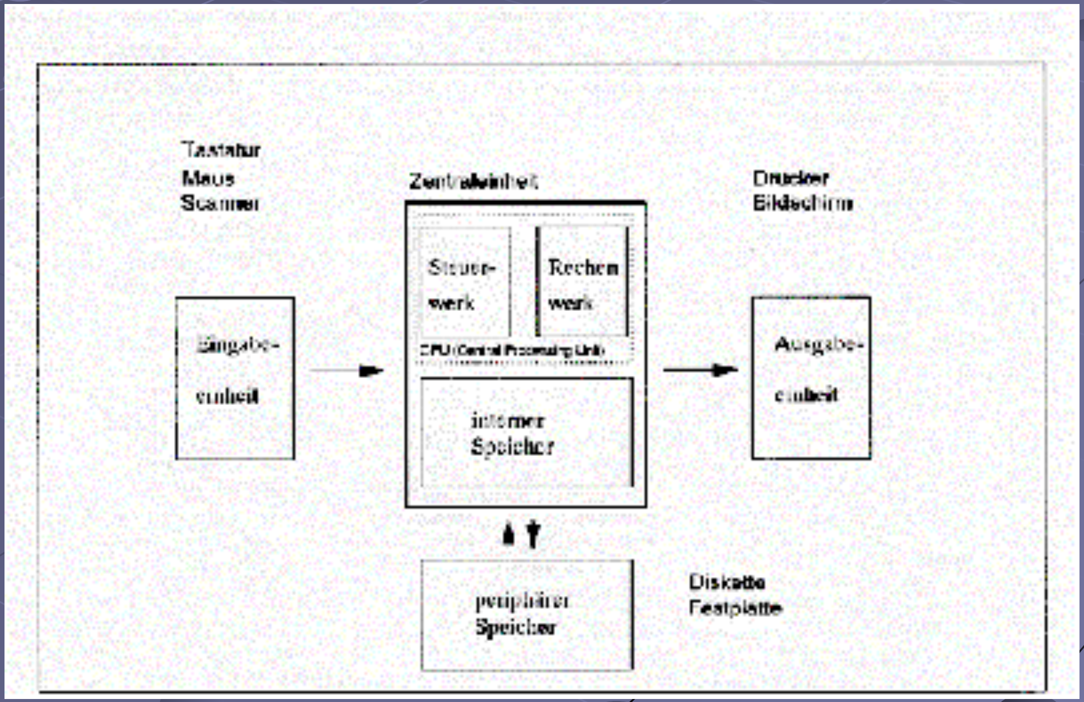
Contact:
Hubert Klahr: klahr@mpia.de

History

- John von Neumann (1903-1957)

Born in Budapest, 1930 Univ. Princeton

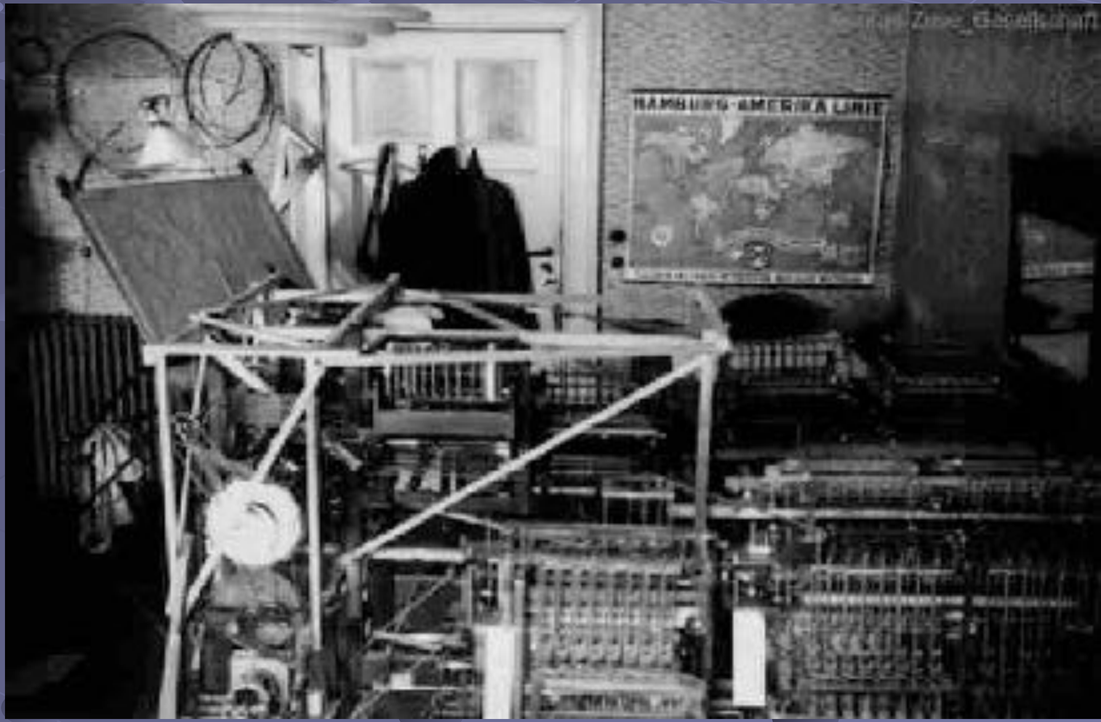
Suggesting an electronic calculation device (1946)



History

- Konrad Zuse (1910-1995) Berlin

Invented the free programmable computer



Z1 in his parents flat: 1936

History

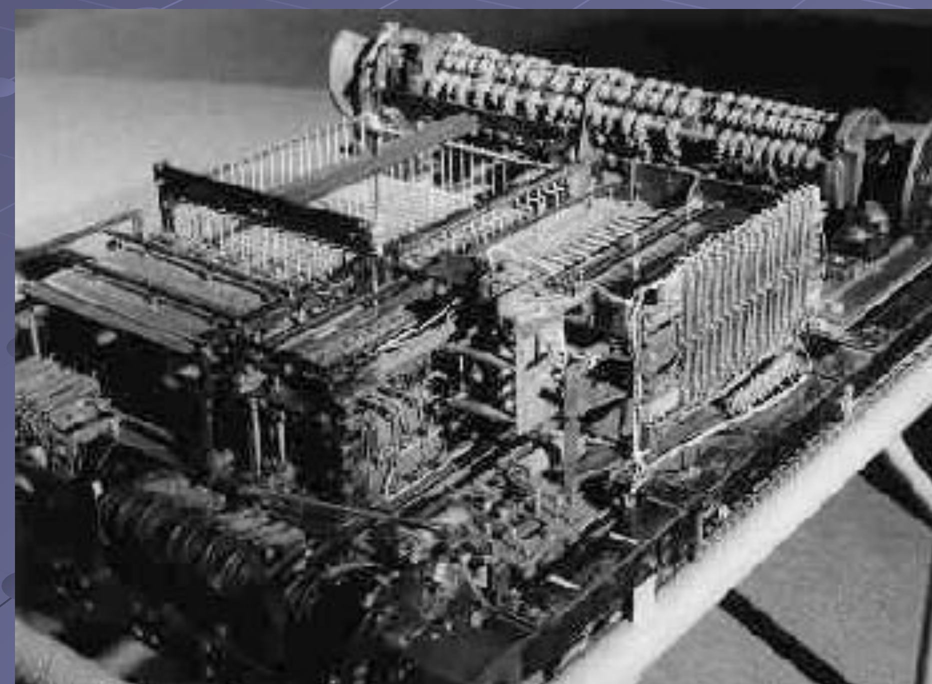
0.03 Mflops

http://www.rtd-net.de/Zuse.html

Zuse Z4: 1944 Berlin, 1950 Zürich

1954 Frankreich

1959 Deutsches Museum München



RAM: 256 byte

Clock Speed: 0.03 MHz

History

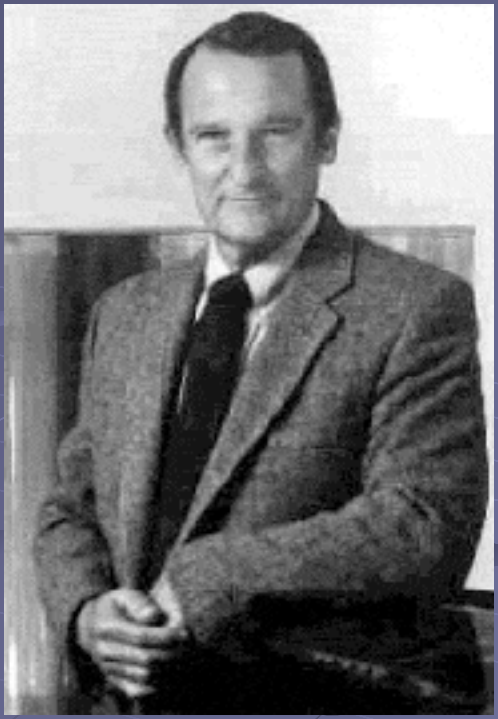
- The principles of electronic computing devices:

Build by Zuse following the theory by von Neumann

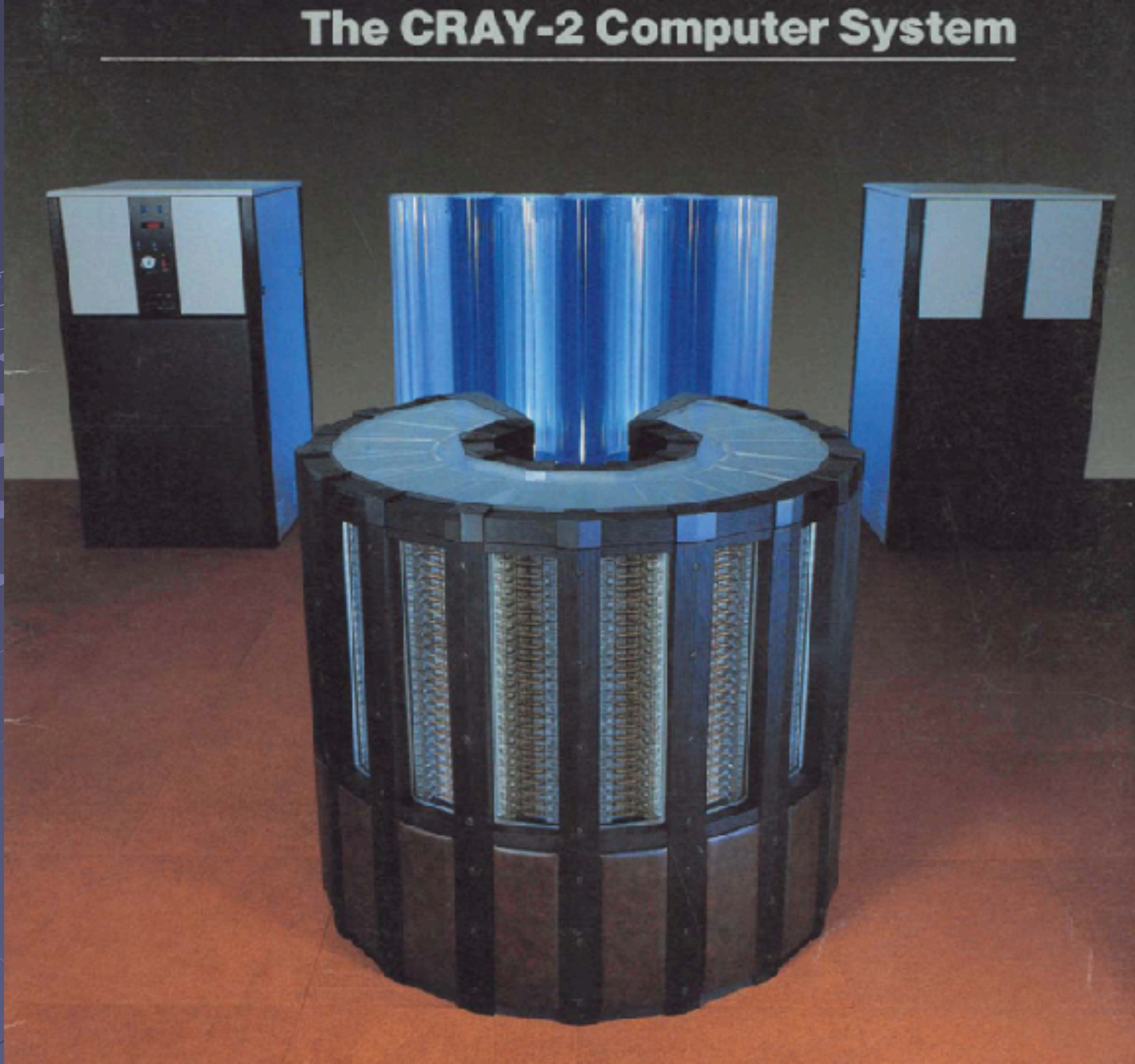
Free programming
Binary representation
Memory
Floating Point Arithmetics

History

- Seymour Cray (1925-1996)
  - ``The father of supercomputing ´´



CRAY1: Vektorregister (1976)
160 Mflop, 80 MHz, 8 MByte RAM
CRAY2: (1984)
1Gflop, 120MHz, 2GByte RAM
iPhone5: 500Mflop... ;)
iPhoneXS: Gflop!


The CRAY-2 Computer System

	Cray 2 (1985)	iPhone XS (2018)	iPhone 13 2021
Price (2017 USD)	> $30,000,000	~$900	$999
Main Processors	4	1 A12X	A15 Bionic chip
Memory (RAM)	256 Megaword	4 Gigabytes	A15 Bionic chip

* megaword -> a 'word' varies but the Cray-2 used 64-bit words with 8 bit parity checks

	Cray 2 (1985)	iPhone XS (2018)	
Storage (max)	~32 GB*	512 GB	1 TB

* Max storage required 32 disk drives of 1.2 GB each

	Cray 2 (1985)	iPhone XS(2018)
Peak Power Consumption	195,000 Watts	< 1 Watt*

*It's very hard to compare peak power given iPhone has so many other functions, so let's just go with < 1 W

	Cray 2 (1985)	iPhone XS(2018)	
Peak Performance	1.9 GFLOP *	~1 GFLOP*	GPU: 1,5 TFLOPS

*GFLOP = Billions of Floating Point operations per second

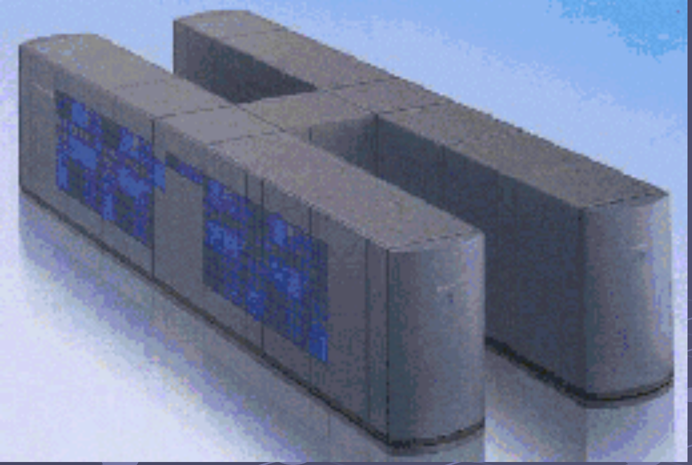
	Cray 2 (1985)	iPhone XS(2018)
Relationship to liquid	Liquid cooled*	Waterproof-ish**

*Cooled with 3M Fluorinert – an electrically inert liquid
** iPhone XS is rated IP68: designed to be waterproof when submerged no deeper than 2 meters (roughly 6 feet) in water for 30 minutes or less.

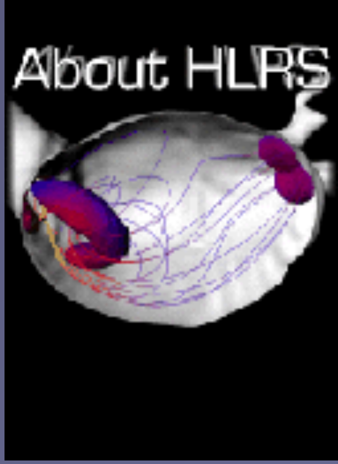
	Cray 2 (1985)	iPhone XS(2018)
Weight	5,500 lb (2,494Kg)	128g (0.3lb, 0.14Kg)
Volume	1.8 cubic meters	~0.007 cubic meters
Height	45 in (1.2 m)	5.8 in (0.16 m)
Width	54 in (1.4 m)*	3.05 in (0.075 m)

* the cray-2 was cylindrical in shape, so the 'width' is really its diameter

ALSO, the iPhone also has 2 cameras, GPS, Celular, Wifi, Bluetooth, A DISPLAY with over 2.7 MM pixels, a battery that lasts a day. A compass. A 3-axis Gyroscope. Speakers. An Accelerometer. Ambient Light and proximity sensors. A Barometer. A Microphone. It can record 4K video. It can take 8MP photos *while* recording 4k video. NFC. iBeacon.



Geschichte



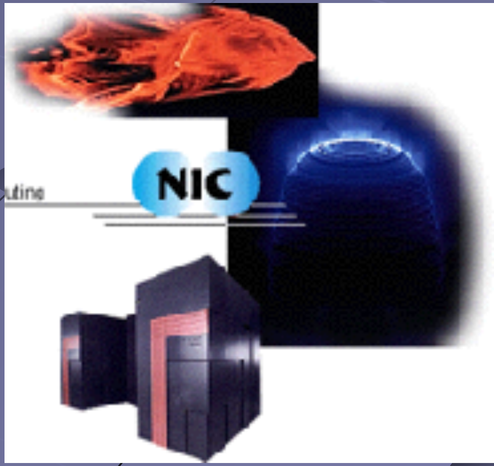
Hitachi SR8000 LRZ München
6 Tflops, TByte Speicher

HLRS Stuttgart

- Wem nutzt es?

- Auto, Luft- und Raumfahrt
- Meteorologie, Klimaforschung, Wetter
- Theoretische Elementarteilchenphysik
- Astrophysik

NIC Jülich



Computer

2008: JUGENE: 294,912 cores; Linpack: 825.5 Teraflops
2013: JUQUEEN: 458,752 cores; Linpack: 5.0 Petaflops
2018: JUWELS: 122,448 cores; 10.4 (CPU) + 1.6 (GPU) Petaflop

Superrechner JUGENE/ JUQUEEN/JUWELS IBM Blue Gene Am FZ Jülich

Eröffnet mit J. Rüttgers Juni 2008

Computer

2008: JUGENE: 294,912 cores; Linpack: 825.5 Teraflops
2013: JUQUEEN: 458,752 cores; Linpack: 5.0 Petaflops
2018: JUWELS: 122,448 cores; 10.4 (CPU) + 1.6 (GPU) Petaflop

Max Planck-Society: Hydra:

3424 compute nodes, 136,960 CPU-cores, 128 Tesla V100-32 GPUs, 240 Quadro RTX 5000 GPUs, 529 TB RAM DDR4, 7.9 TB HBM2,

11.4 PFlop/s peak DP, 2.64 PFlop/s peak SP

Max Planck-Society: Raven:

1592 CPU compute nodes, 114624 CPU cores, 421 TB DDR RAM, 8.8 PFlop/s theoretical peak performance (FP64), plus 192 GPU-accelerated compute nodes 768 GPUs, 30 TB HBM2,

14.6 PFlop/s theoretical peak performance (FP64).

Computer

2007…
GeForce 8800 GTX, 128 Stream Proc., 768 MB
GeForce 8800 GTS, 128 Stream Proc., 512 MB
GeForce 8800 GT, 112 Stream Proc., 512 MB

2008…
GeForce 9800 GTX, 128 Stream Proc., 512 MB
GeForce 9800 GX2, 256 Stream Proc., 1 GB
GeForce 9800 GT, 64 Stream Proc., 512 MB

http://www.nvidia.com

Graphic Cards (GPU) …

Introduction to Scientific Computing

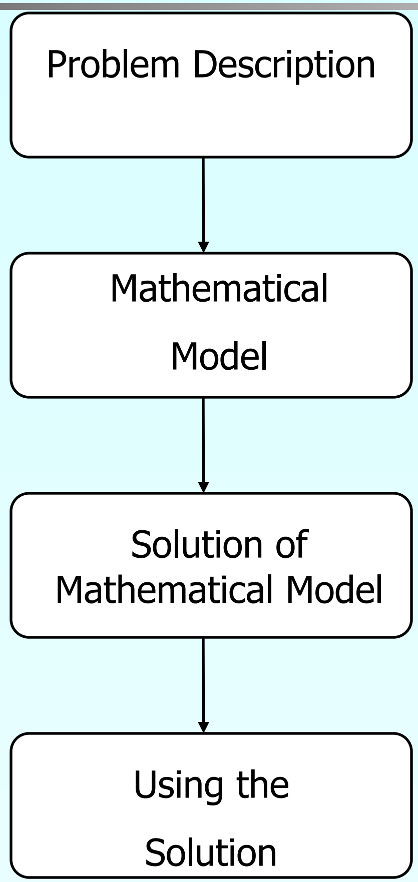
Major: All Engineering Majors

Authors: Autar Kaw, Luke Snyder

http://numericalmethods.eng.usf.edu

Numerical Methods for STEM undergraduates

How do we solve an engineering problem?

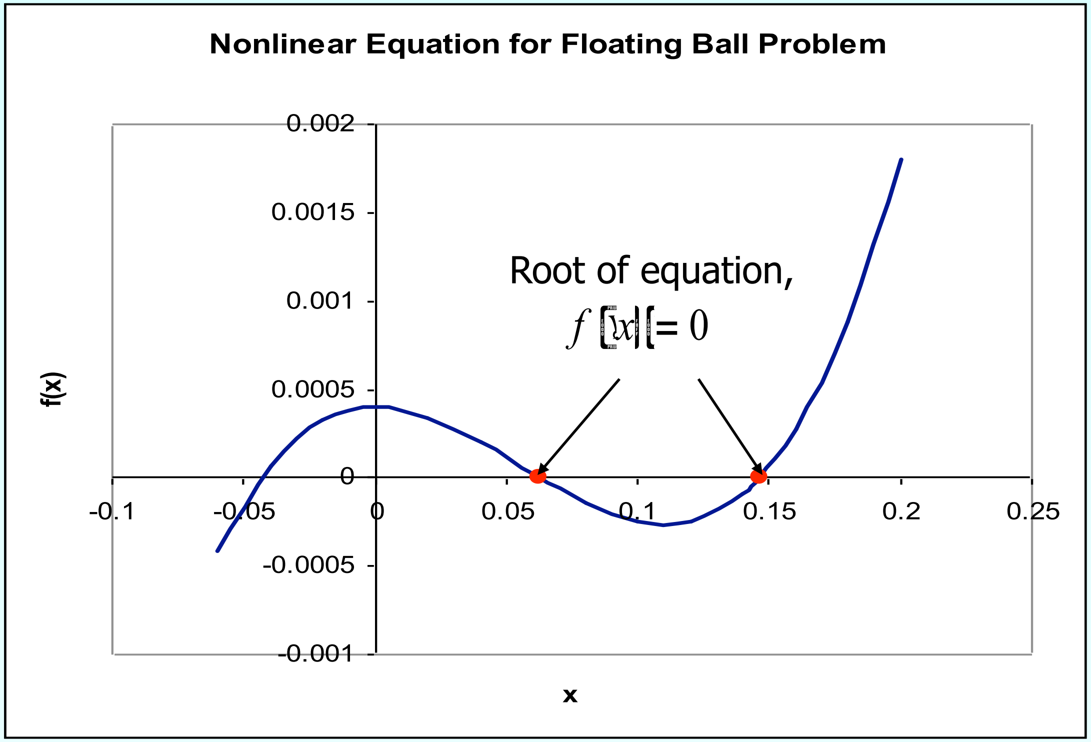


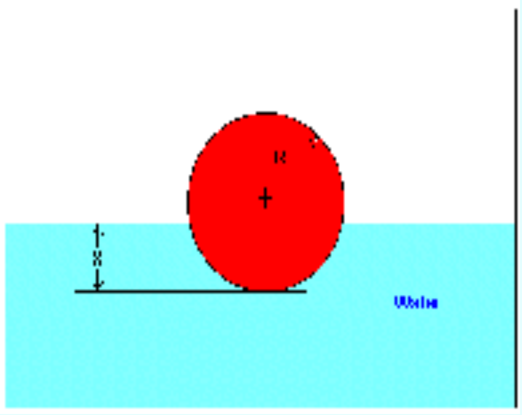
Mathematical Procedures

- Nonlinear Equations
- Differentiation
- Simultaneous Linear Equations
- Curve Fitting
  - Interpolation
  - Regression
- Integration
- Ordinary Differential Equations
- Other Advanced Mathematical Procedures:
  - Partial Differential Equations
  - Optimization
  - Fast Fourier Transform

Nonlinear Equations

Floating Ball Problem

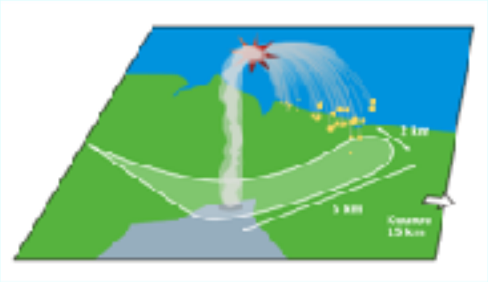


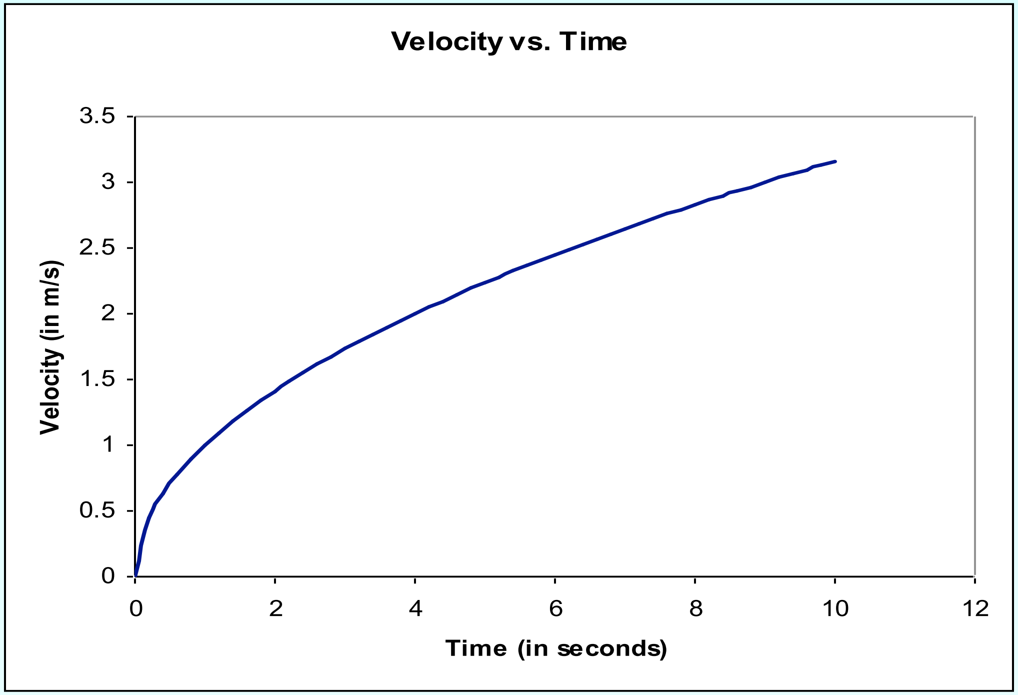


$$f(x) = x^3 - 0.165x^2 + 3.993 \times 10^{-4}$$

Differentiation

Velocity vs. time rocket problem





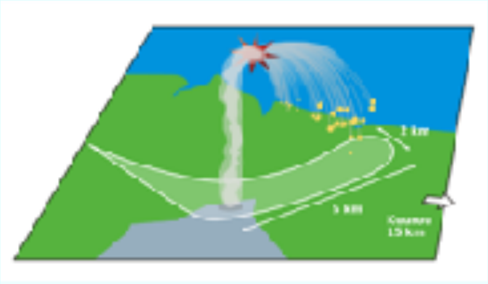
$$a = \frac{dv}{dt}$$

What is the acceleration at t=10 seconds?

Simultaneous Linear Equations

Find the velocity profile from

Time,t	Velocity,v
s	m/s
5	106.8
8	177.2
12	279.2



$$v(t) = at^2 + bt + c$$

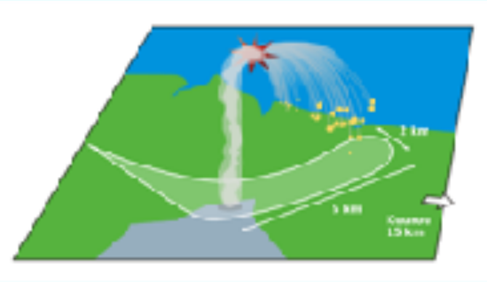
$$5 \le t \le 12$$

Three simultaneous linear equations:

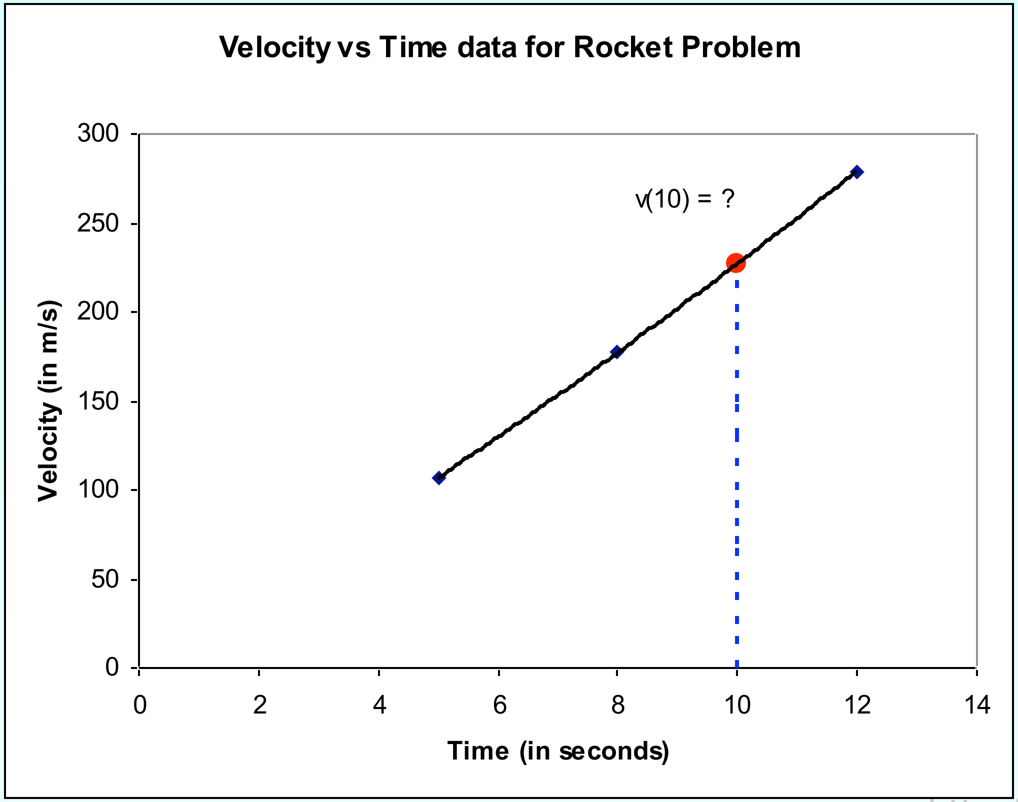
$$25a + 5b + c = 106.8$$
$$64a + 8b + c = 177.2$$
$$144a + 12b + c = 279.2$$

Interpolation

What is the velocity of the rocket at t=10 seconds?

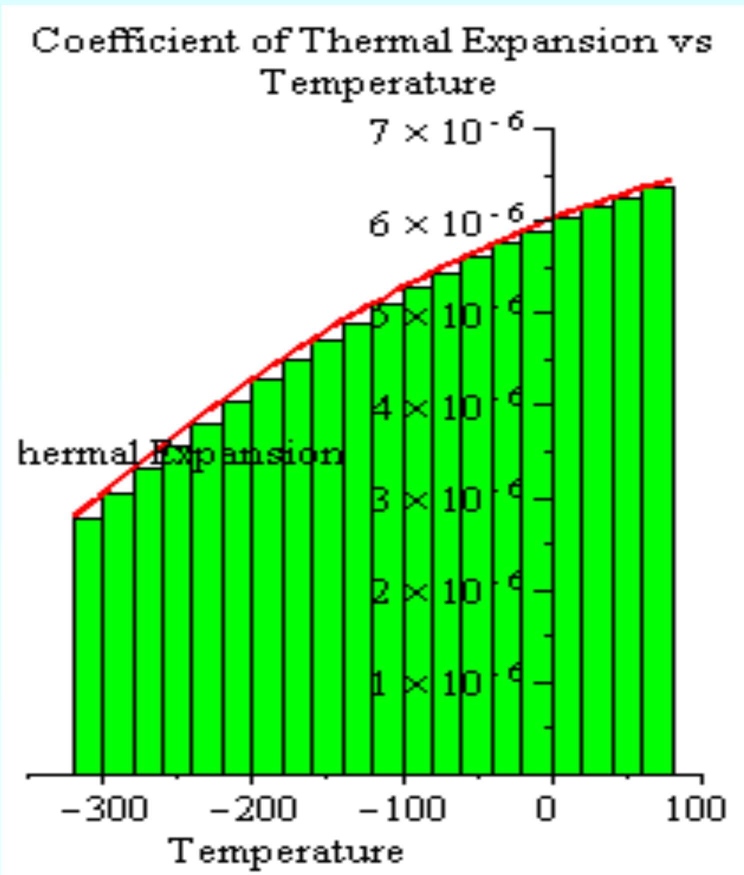


Time,t	Velocity,v
s	m/s
5	106.8
8	177.2
12	279.2



Integration

Finding the contraction in a metal construction part.



$$\text{á} = a_0 + a_1 T + a_2 T^2$$
$$= 6.0217\times10^{-6} + 6.2782\times10^{-9}\,\mathrm{T} - 1.2218\times10^{-11}\,\mathrm{T}^2$$

$$\Delta D = D \int_{T_{room}}^{T_{fluid}} \alpha \, dT$$

Ex: magnetohydrodynmical equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\mathbf{v}\,\rho) = 0$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} - \mathbf{B}\mathbf{B}) + \nabla p_* = -\rho \mathbf{g}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\mathbf{v}\,(\rho E + p_*) - \mathbf{B}\,(\mathbf{v} \cdot \mathbf{B})) = \rho \mathbf{g} \cdot \mathbf{v} + \Gamma - \Lambda$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla \cdot (\mathbf{v}\mathbf{B} - \mathbf{B}\mathbf{v}) = 0$$

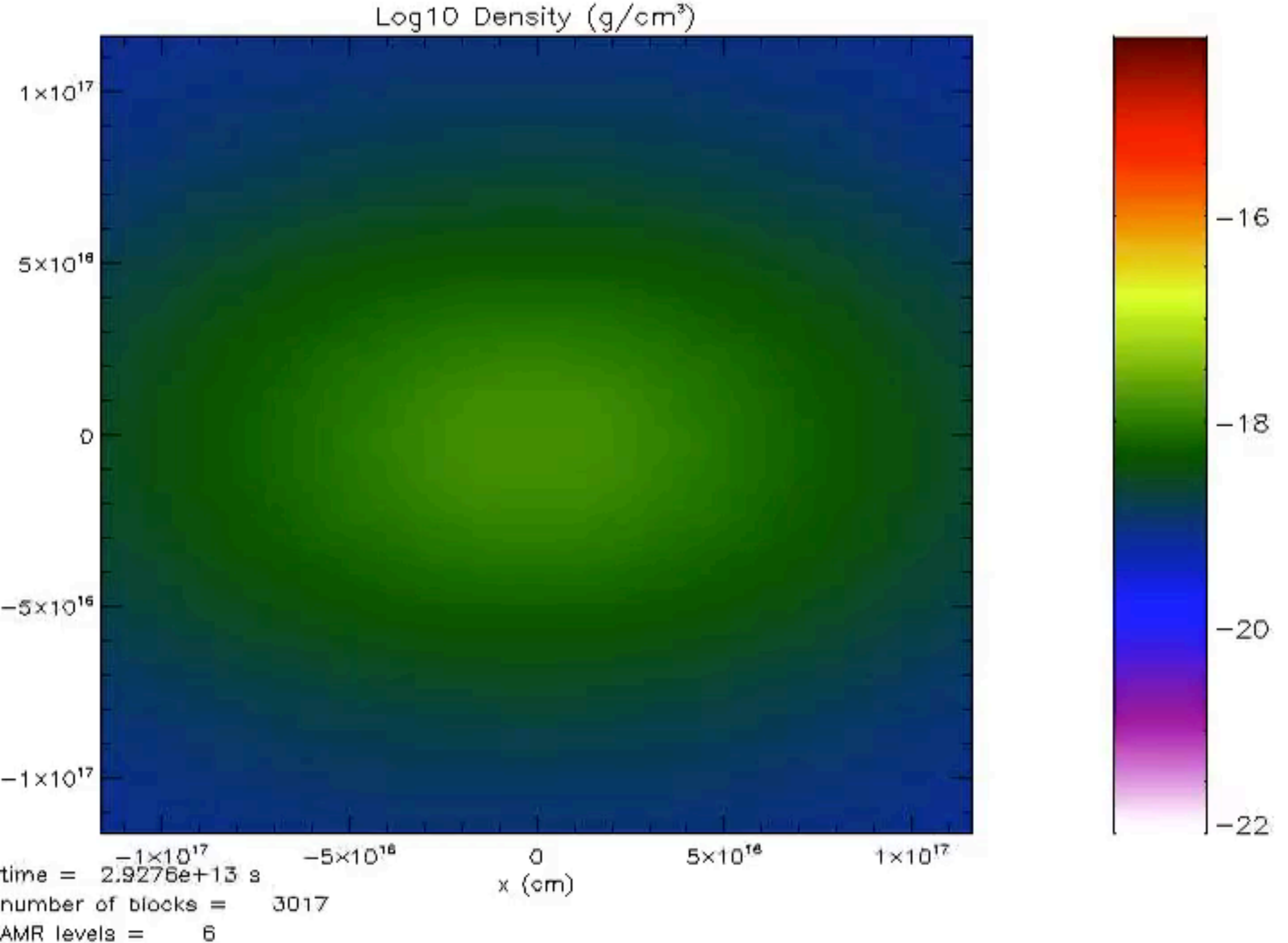
$$E = \frac{1}{2} v^2 + \varepsilon + \frac{1}{2}\frac{B^2}{\rho},$$

$$p_* = p + \frac{B^2}{2},$$

$$p = (\gamma - 1)\,\rho\varepsilon$$

$$\mathbf{g} = -\nabla \Phi \qquad \Delta \Phi = 4\pi G \rho$$

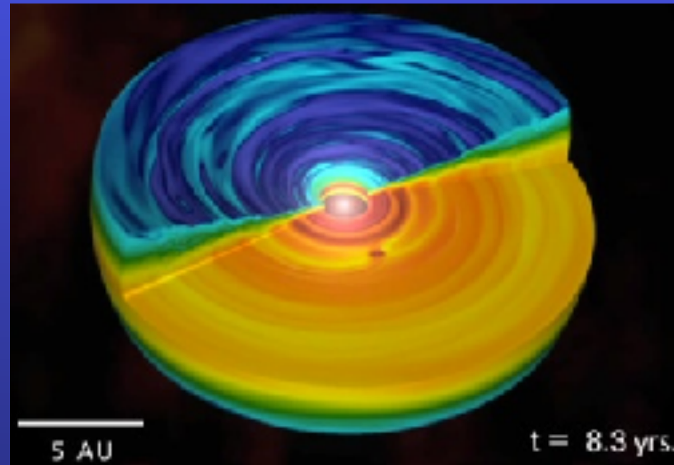
Ideal MHD + self-gravity + ideal gas + heating & cooling

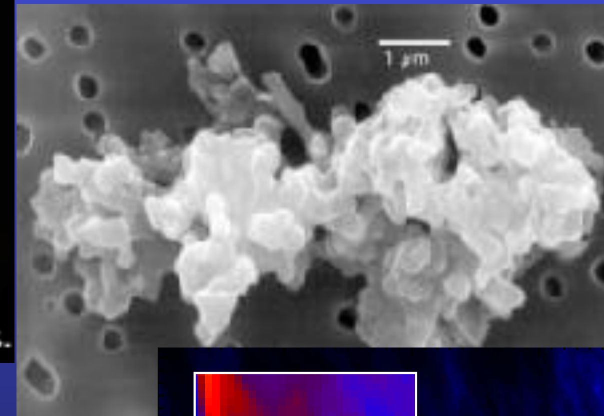

Log10 Density (g/cm³)
1×10¹⁷
5×10¹⁶
0
−5×10¹⁶
−1×10¹⁷
−1×10¹⁷
−5×10¹⁶
0
5×10¹⁶
1×10¹⁷
x (cm)
−16
−18
−20
−22
time = 2.9276e+13 s
number of blocks = 3017
AMR levels = 6

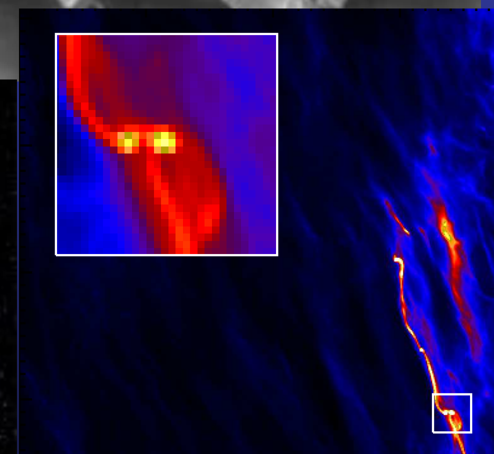
Ex: magnetohydrodynmical equations

with random component: $B_x = 3\mu G + \delta b = 3\mu G$

Garching, Feb 1st, 2011







The nature and role of Turbulence in Planet Formation:

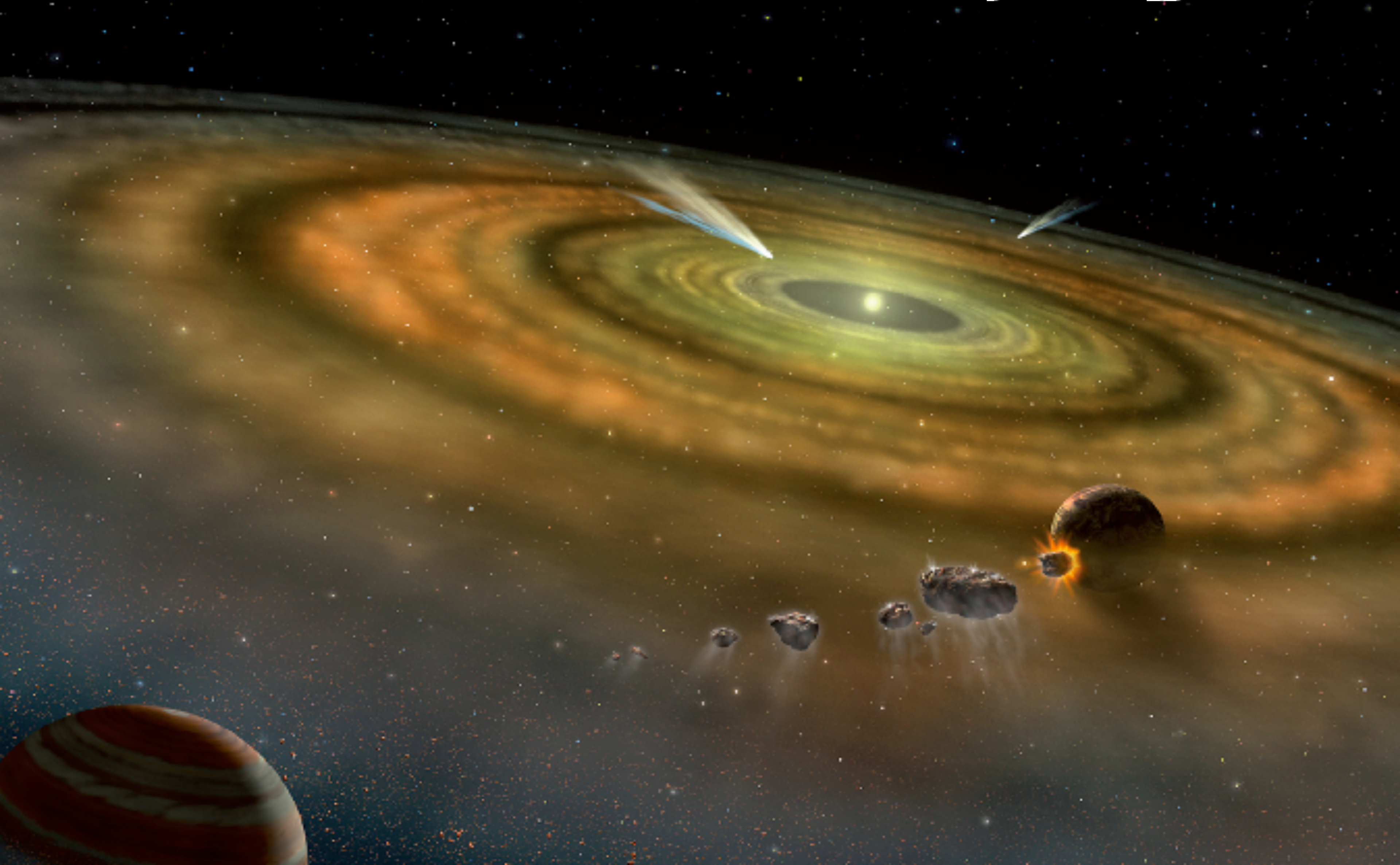
Magnetorotational and Baroclinic Instability.

Hubert Klahr,

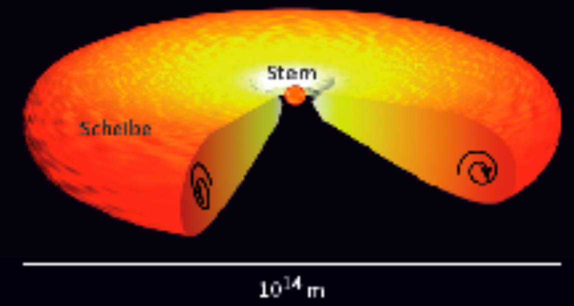
Max-Planck-Institut für Astronomie, Heidelberg

Wlad Lyra (AMNH), Peter Bodenheimer (Santa Cruz), Anders Johansen (Lund), Natalia Raettig, Helen Morrison, Mario Flock, Natalia Dzyurkevich, Karsten Dittrich, Til Birnstiel, Kees Dullemond, Chris Ormel (MPIA), Neal Turner (JPL), Jeffrey S. Oishi (Berkley), Mordecai-Mark Mac Low (AMNH), Andrew Youdin (CITA), Doug Lin (Santa Cruz)

"Birth places of Planets:"
Gas and dust disks around young stars



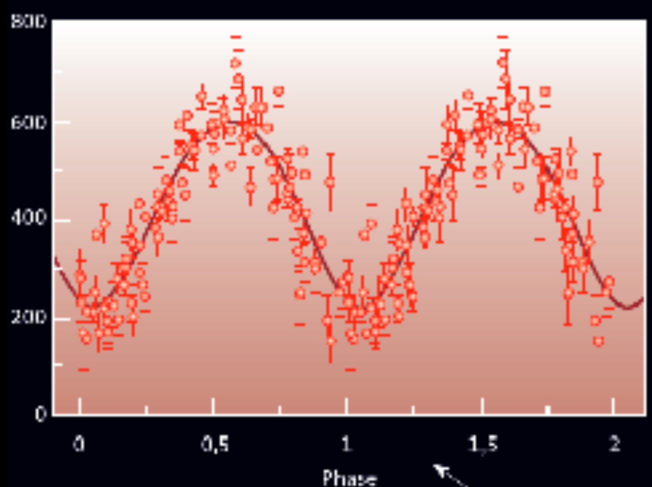
$t = 0$

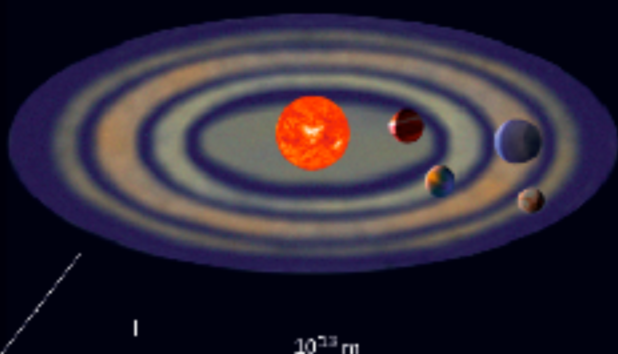


(a)

... a miracle occurs ...

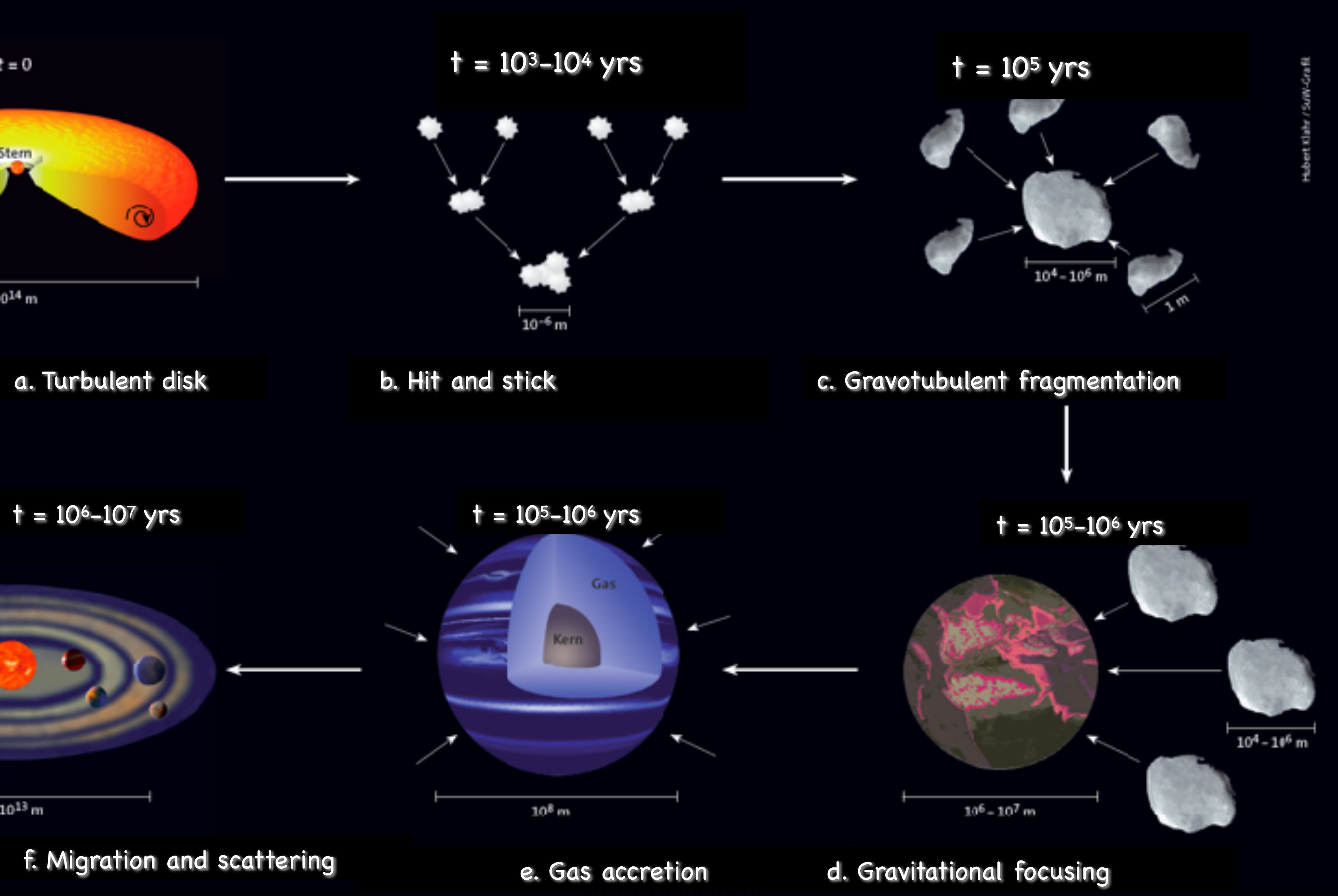
$t = 10^7$





(f)

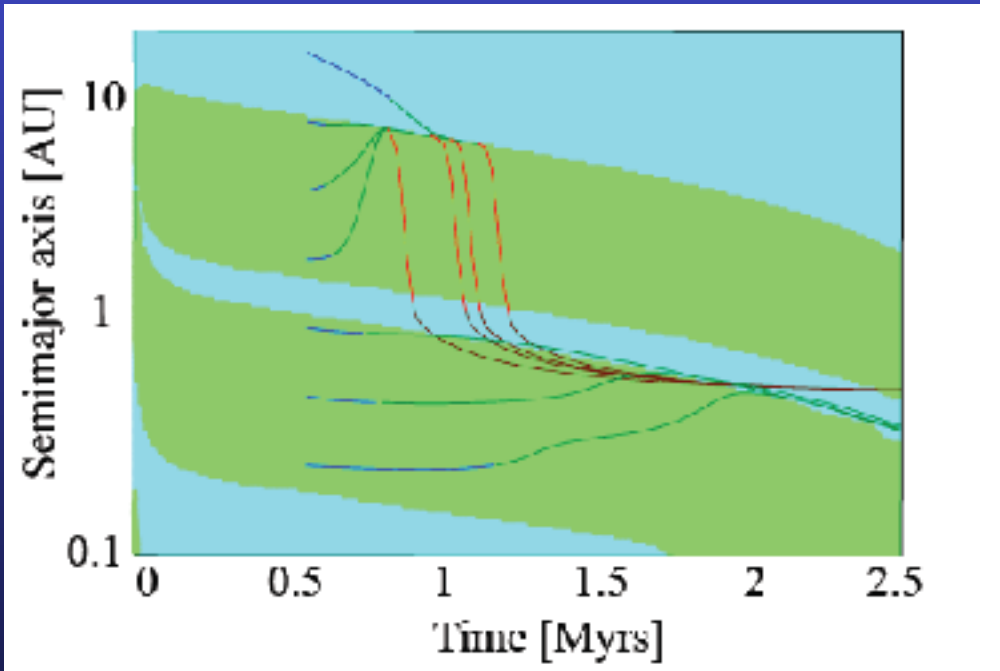
The planetary construction plant.



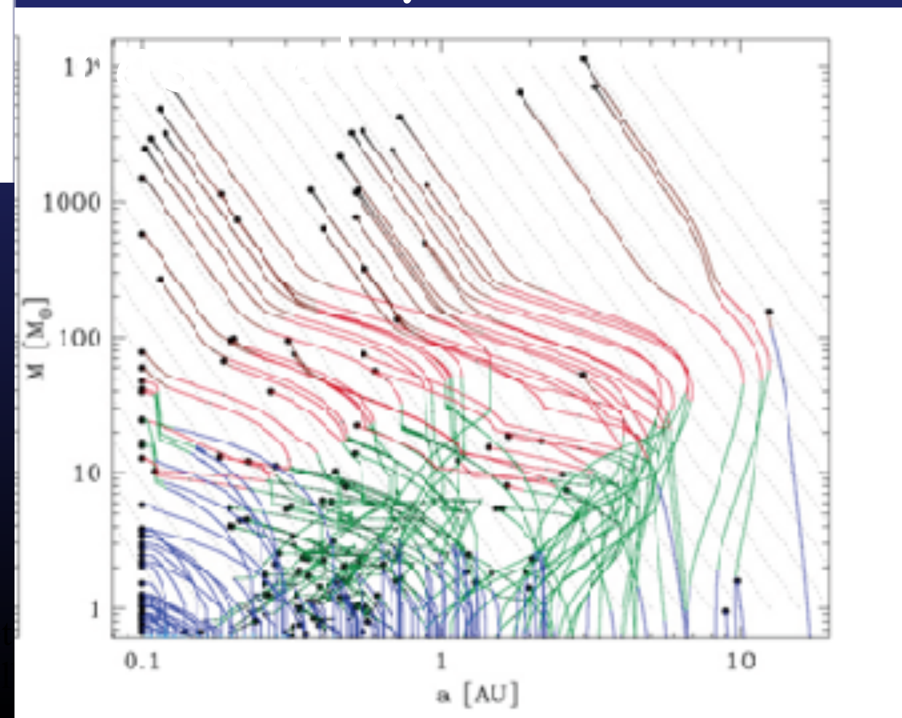
Synthetic Populations...

Application of recent results on the orbital migration of low mass planets in planetary population synthesis

C. Mordasini[1], K.-M. Dittkrist[1], Y. Alibert[2], H. Klahr[1], W. Benz[2] and T. Henning[1]
[1]Max Planck Institute for Astronomy, Konigstuhl 17, D-69117 Heidelberg, Germany
email: mordasini@mpia.de
[2]Physikalisches Institut, Sidlerstrasse 5, CH-3012 Bern, Switzerland



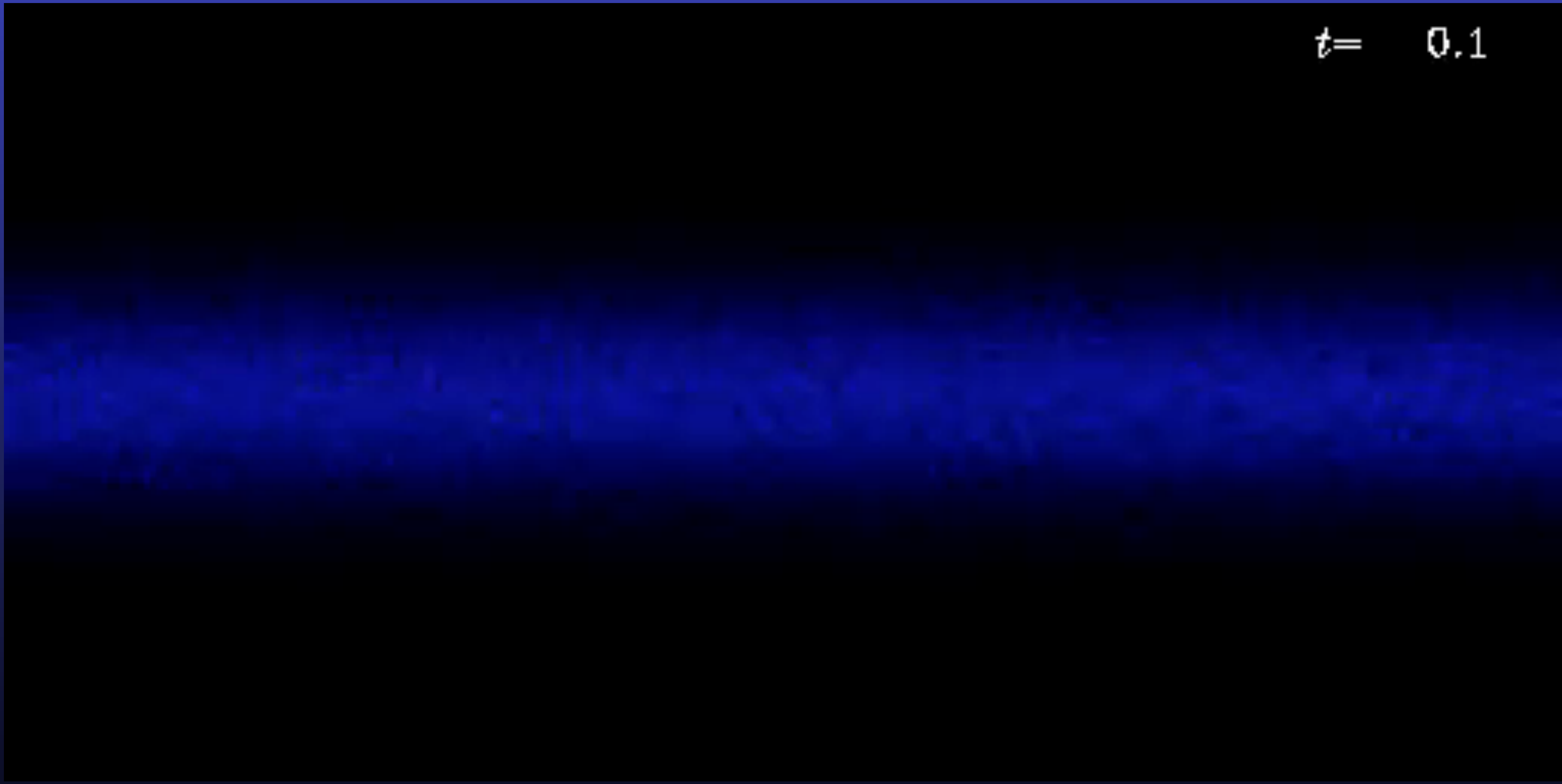
...to explore the importance of metallicity, stellar



...and to test the individual modeling steps of planet formation by comp. To observations.

10 cm sized boulders:

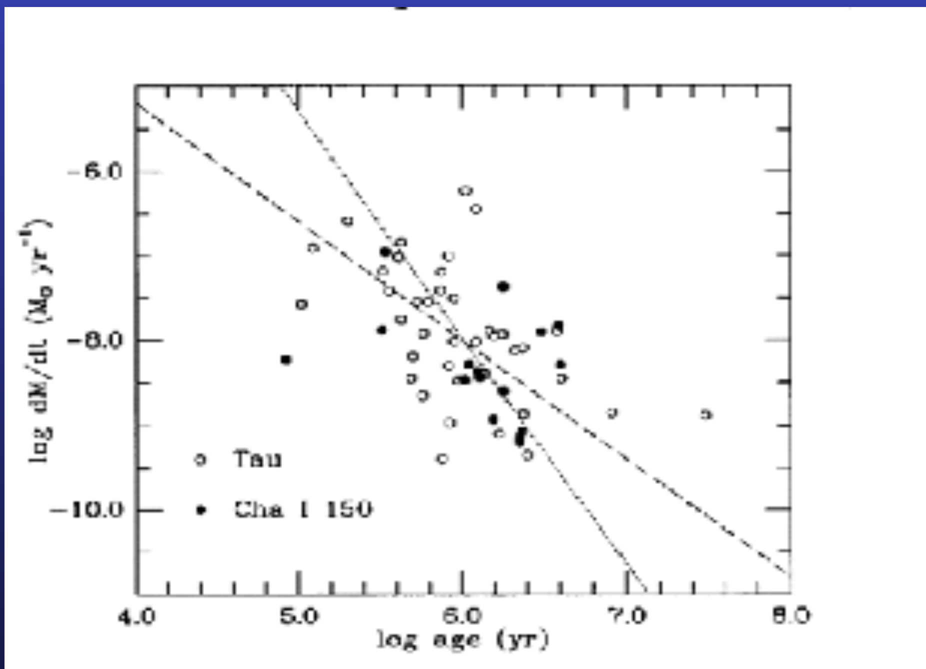
V e r t i c a l

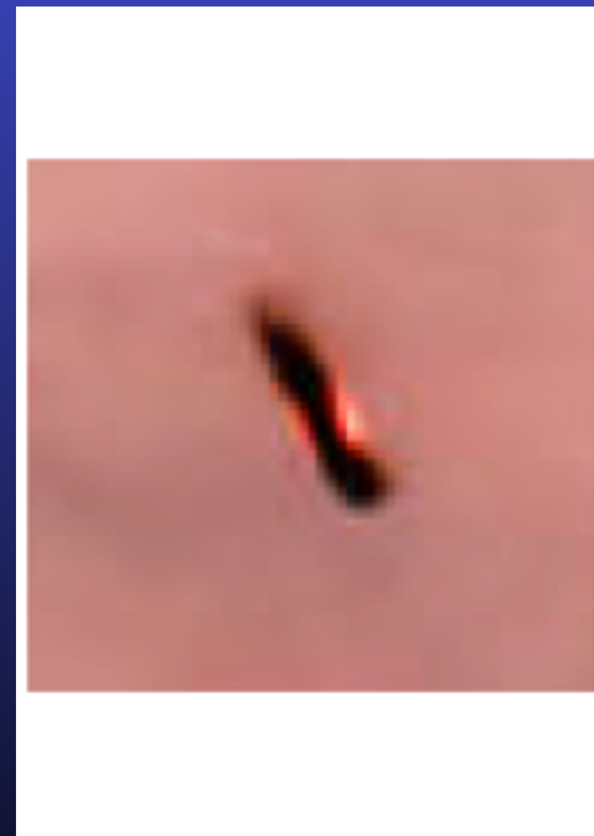


h o r i z o n t a l

Johansen, Henning & Klahr 2006

Accretion Energy in rotating systems => Turbulent transport of angular momentum

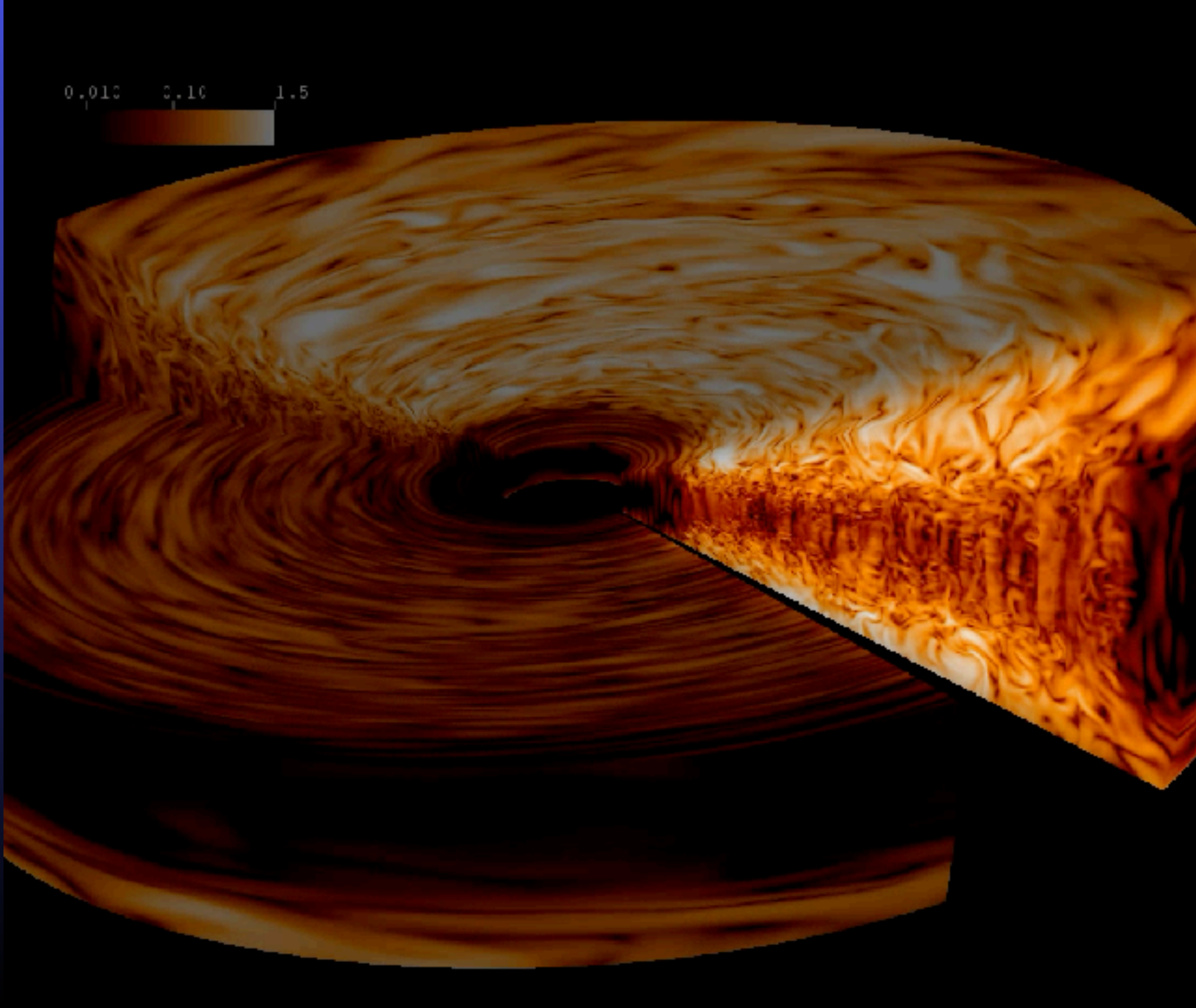




Hartmann et al. 1998, 2006

alpha = 0.01

WHY DO T TAURI DISKS ACCRETE?


0.010
0.10
1.5

Pluto Code: HLLD Upwind CT, piecewise linear reconstruction, Runge Kutta 2nd order

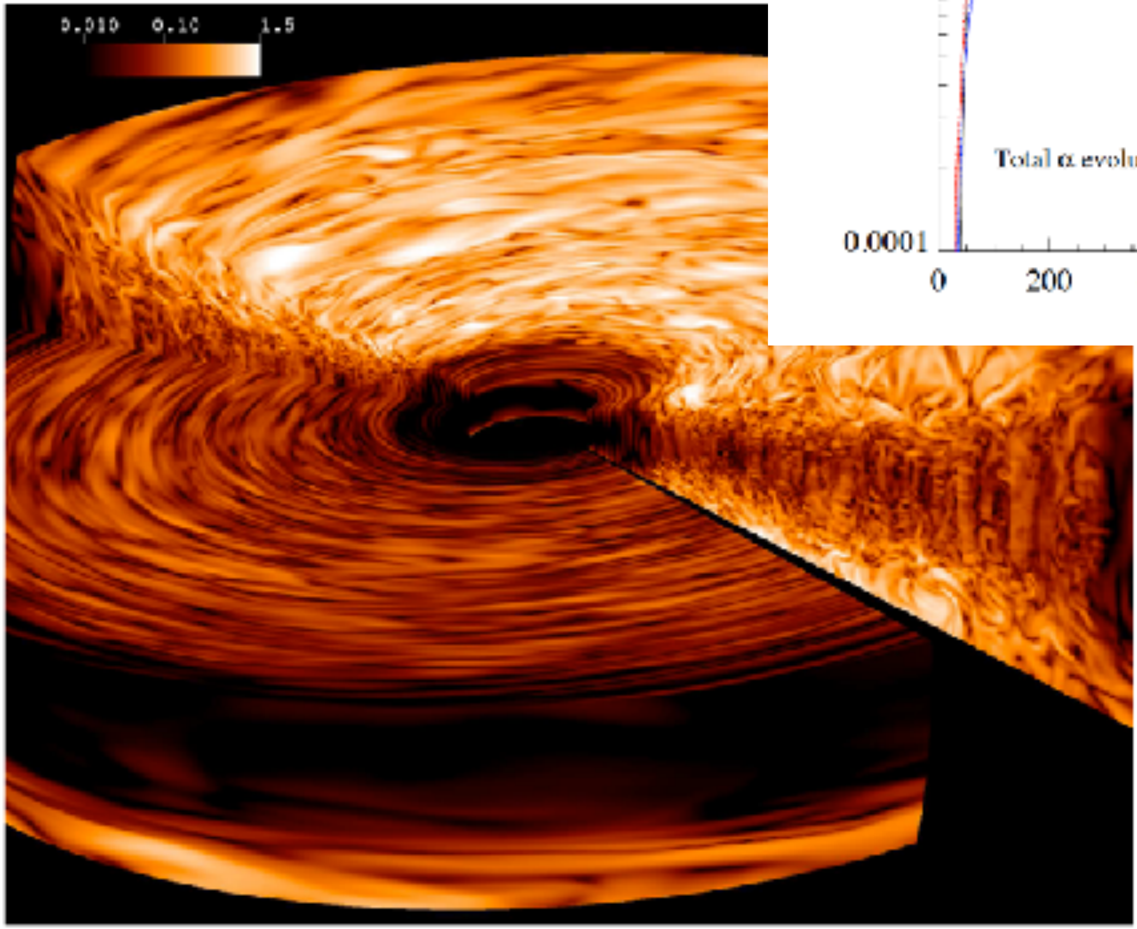
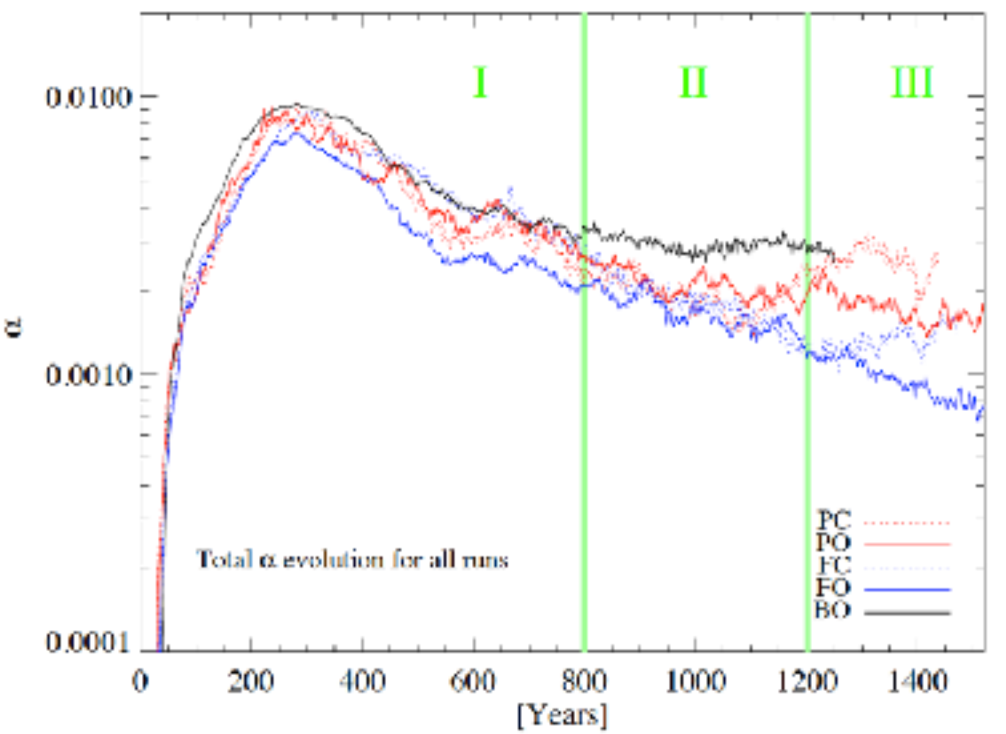


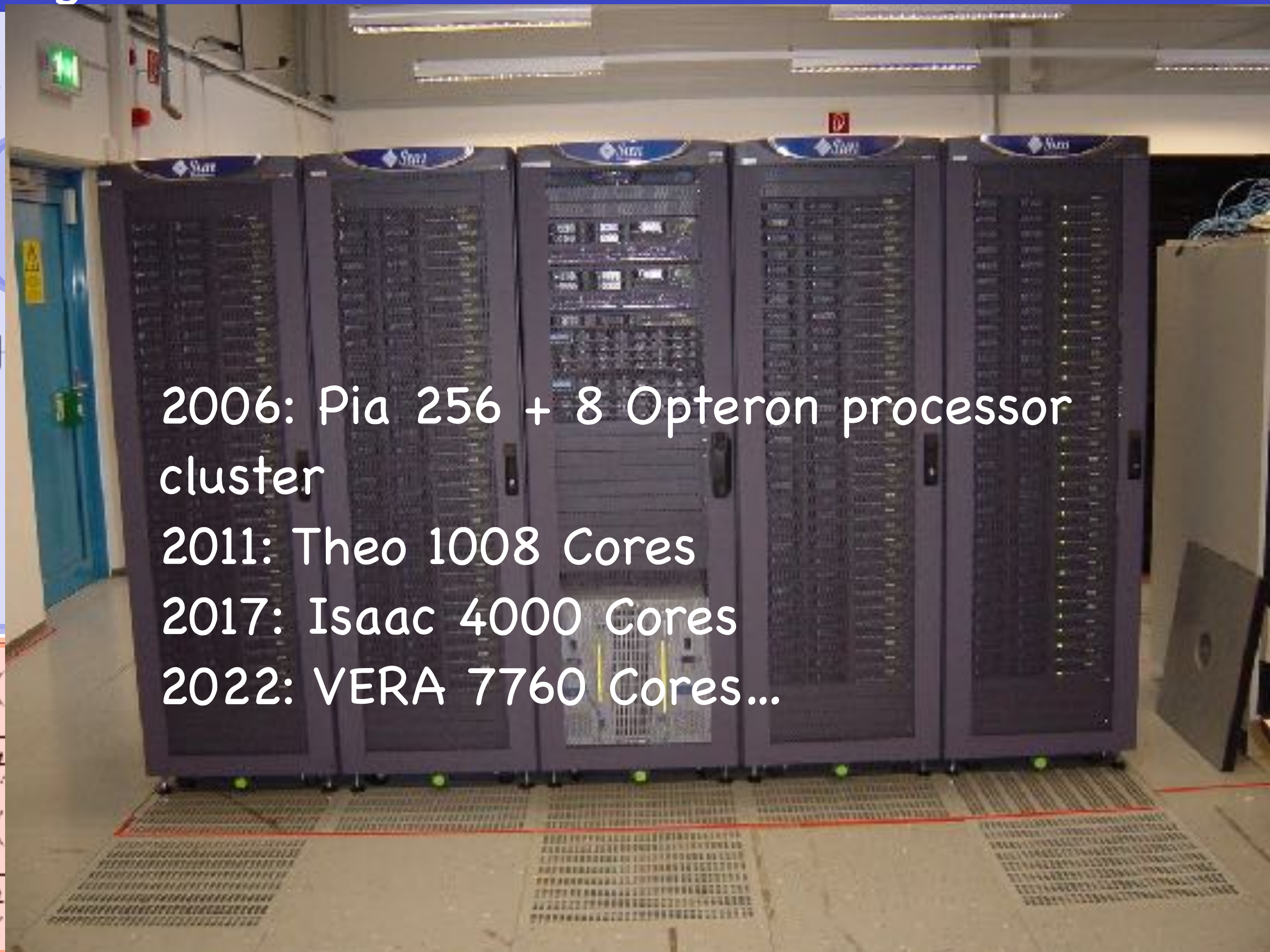
Fig. 5.— 3D contour plot of turbulent rms velocity at 750 inner orbits for model BO.



384x192x768

Global 360 stratified!
At 20 grid cells per H!
1.8 Million CPU hours

MRI plus self-gravity for the dust, including particle feed back on the gas: collaboration with Mac Low & Oichi AMNH

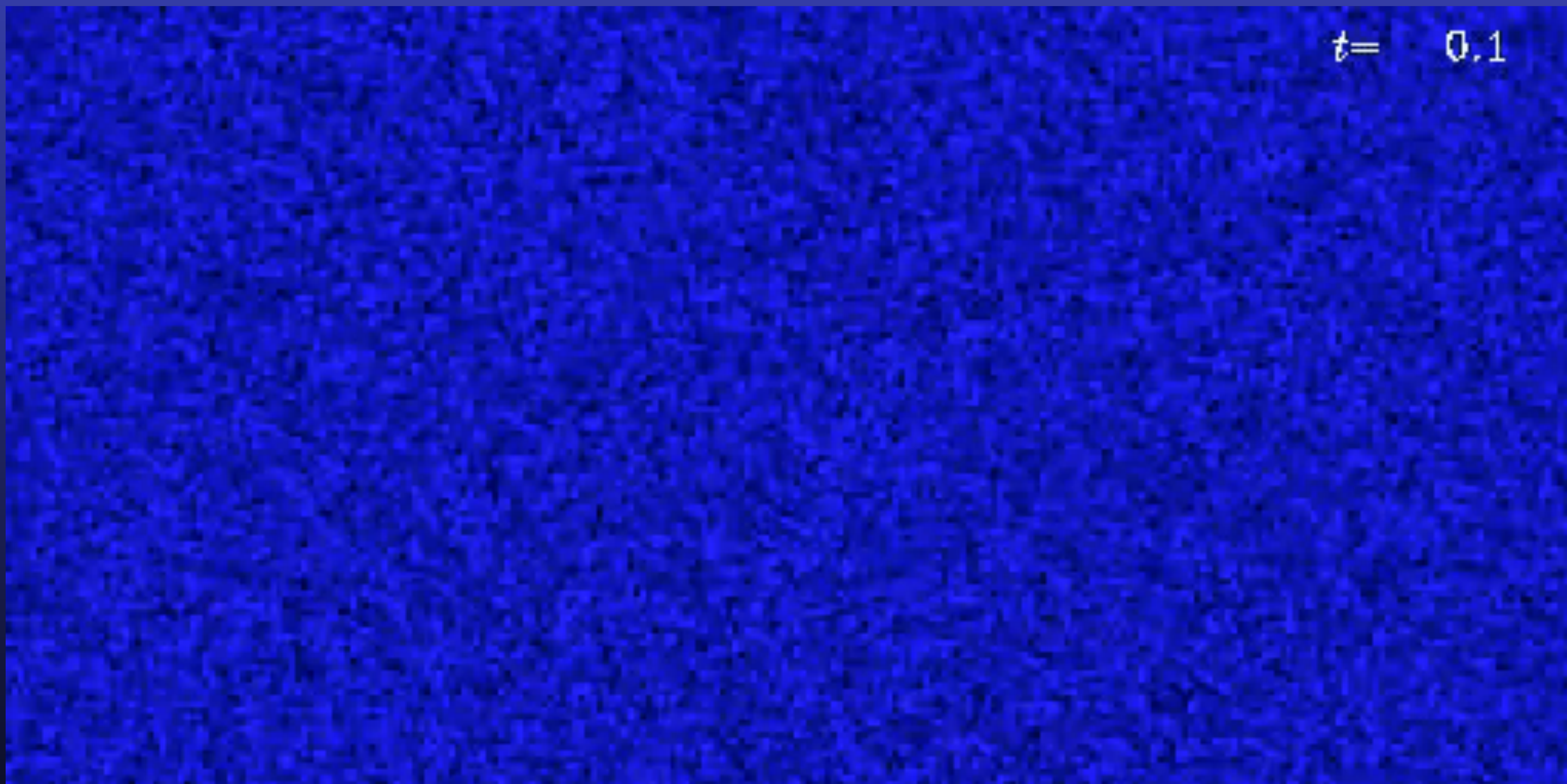


2006: Pia 256 + 8 Opteron processor cluster
2011: Theo 1008 Cores
2017: Isaac 4000 Cores
2022: VERA 7760 Cores...

Poisson equation solved via FFT in parallel mode: up to 256^3 cells

Streaming instability for radial drift:

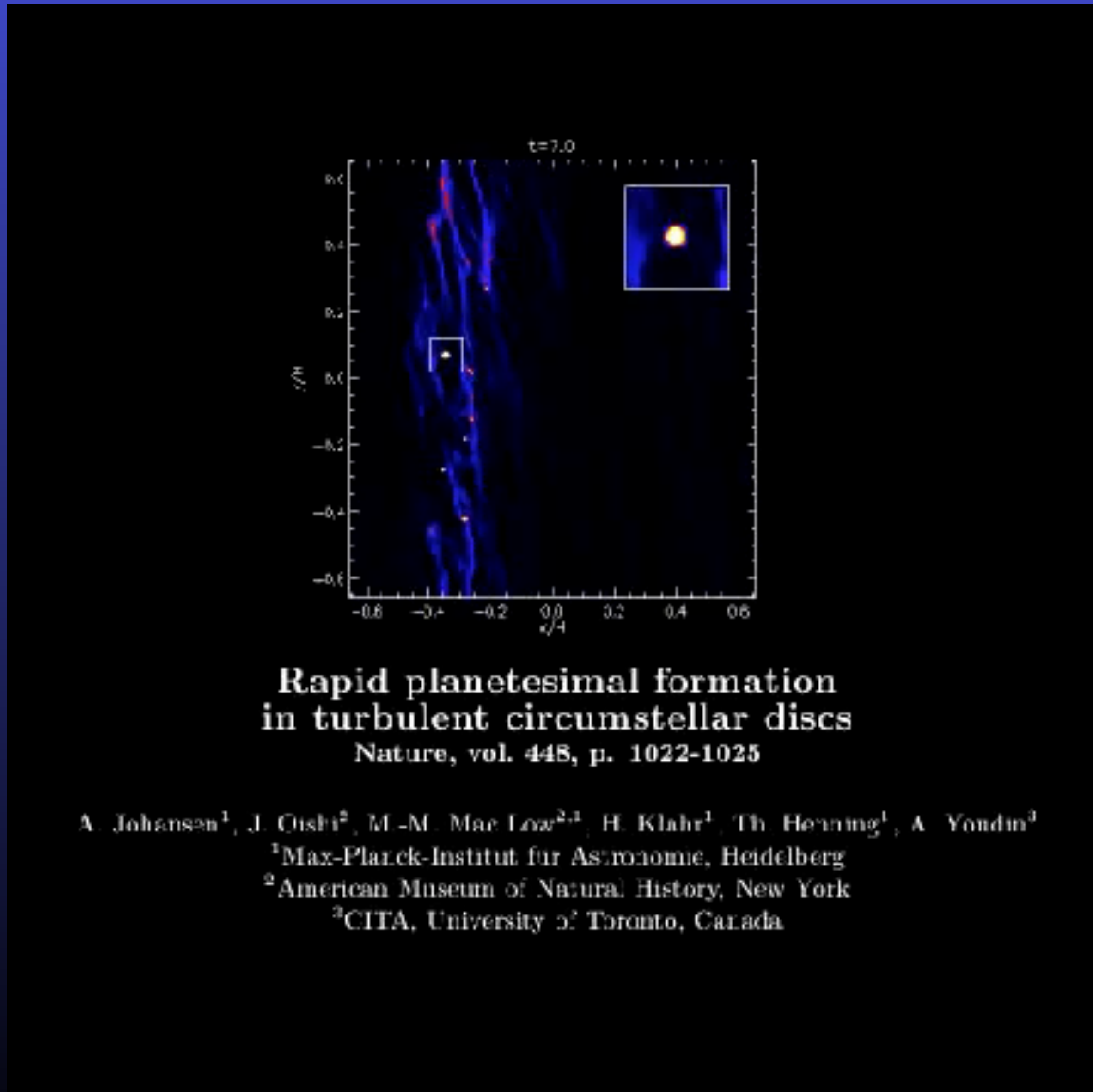
ve
r
t
i
ca
l



r a d i a l

This is what laminar radial drift actually looks like!

Johansen, Oichi, MacLow, Klahr, Henning & Youdin, 2007, nature



http://www.mpia.de/homes/johansen/research_en.php

Binary Representation

Major: All Engineering Majors

Authors: Autar Kaw, Matthew Emmons

http://numericalmethods.eng.usf.edu

Numerical Methods for STEM undergraduates

How a Decimal Number is Represented

$$257.76 = 2\times10^{2} + 5\times10^{1} + 7\times10^{0} + 7\times10^{-1} + 6\times10^{-2}$$

Base 2

$$(1011.0011)_2 = \begin{pmatrix} (1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0) \\ + (0\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3} + 1\times 2^{-4}) \end{pmatrix}_{10}$$

$$= 11.1875$$

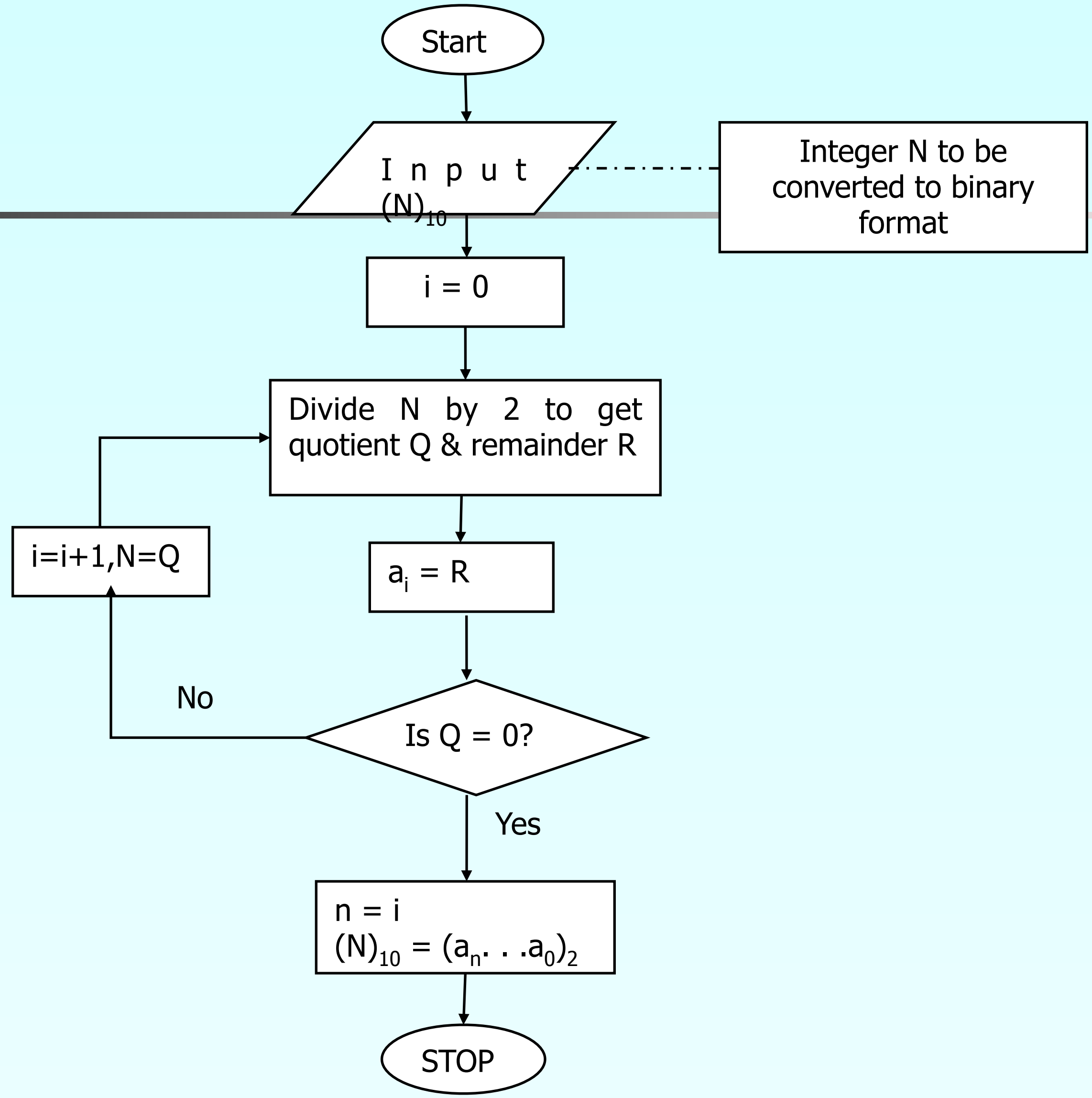
Convert Base 10 Integer to binary representation

Table 1 Converting a base-10 integer to binary representation.

	Quotient	**Remainder**
11/2	5	$1 = a_0$
5/2	2	$1 = a_1$
2/2	1	$0 = a_2$
1/2	0	$1 = a_3$

Hence

$$(11)_{10} = (a_3 a_2 a_1 a_0)_2$$
$$= (1011)_2$$


Start
Input
(N)10
Integer N to be converted to binary format
i = 0
Divide N by 2 to get quotient Q & remainder R
i=i+1,N=Q
ai = R
No
Is Q = 0?
Yes
n = i
(N)10 = (an. . .a0)2
STOP

Fractional Decimal Number to Binary

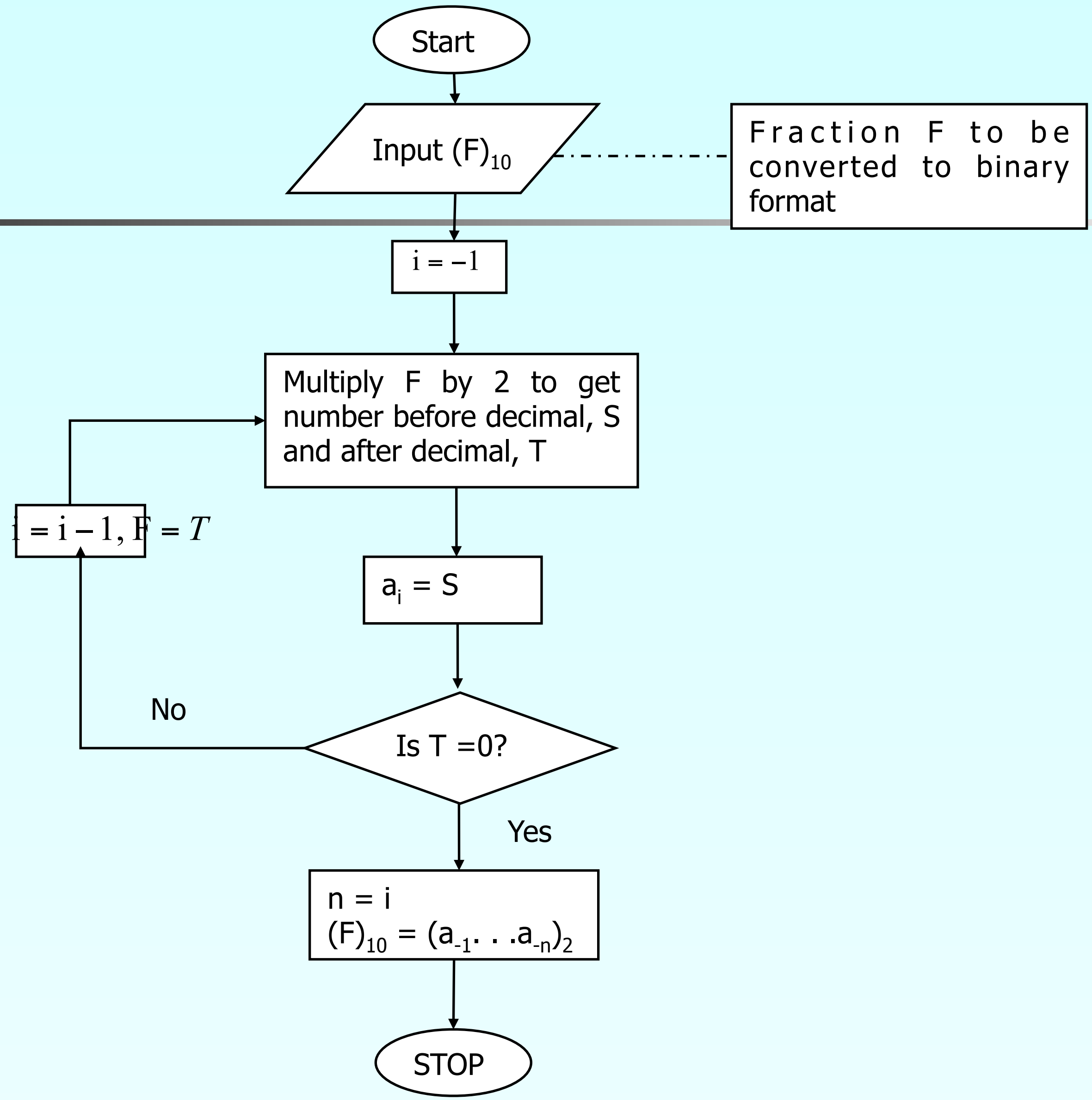
Table 2. Converting a base-10 fraction to binary representation.

	Number	Number after decimal	Number before decimal
0.1875×2	0.375	0.375	$0 = a_{-1}$
0.375×2	0.75	0.75	$0 = a_{-2}$
0.75×2	1.5	0.5	$1 = a_{-3}$
0.5×2	1.0	0.0	$1 = a_{-4}$

Hence

$$(0.1875)_{10} = (a_{-1}a_{-2}a_{-3}a_{-4})_2$$

$$= (0.0011)_2$$


Start
Input (F)10
Fraction F to be converted to binary format
i = −1
Multiply F by 2 to get number before decimal, S and after decimal, T
i = i − 1, F = T
a_i = S
No
Is T =0?
Yes
n = i
(F)10 = (a-1. . .a-n)2
STOP

Decimal Number to Binary

$$(11.1875)_{10} = (\ \ ?.?\ \ )_2$$

Since

$$(11)_{10} = (1011)_2$$

and

$$(0.1875)_{10} = (0.0011)_2$$

we have

$$(11.1875)_{10} = (1011.0011)_2$$

All Fractional Decimal Numbers Cannot be Represented Exactly

Table 3. Converting a base-10 fraction to approximate binary representation.

	Number	**Number after decimal**	**Number before Decimal**
0.3×2	0.6	0.6	$0 = a_{-1}$
0.6×2	1.2	0.2	$1 = a_{-2}$
0.2×2	0.4	0.4	$0 = a_{-3}$
0.4×2	0.8	0.8	$0 = a_{-4}$
0.8×2	1.6	0.6	$1 = a_{-5}$

$$(0.3)_{10} \approx (a_{-1}a_{-2}a_{-3}a_{-4}a_{-5})_2 = (0.01001)_2 = 0.28125$$

Another Way to Look at Conversion

Convert $(11.1875)_{10}$ to base 2

$$
\begin{aligned}
(11)_{10} &= 2^3 + 3 \\
&= 2^3 + 2^1 + 1 \\
&= 2^3 + 2^1 + 2^0 \\
&= 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0 \\
&= (1011)_2
\end{aligned}
$$

$$(0.1875)_{10} = 2^{-3} + 0.0625$$

$$= 2^{-3} + 2^{-4}$$

$$= 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$$

$$= (.0011)_2$$

$$(11.1875)_{10} = (1011.0011)_2$$

Floating Point Representation

Major: All Engineering Majors

Authors: Autar Kaw, Matthew Emmons

http://numericalmethods.eng.usf.edu

Numerical Methods for STEM undergraduates

Floating Decimal Point – Scientific Form

$$256.78 \text{ is written as } +2.5678 \times 10^{2}$$

$$0.003678 \text{ is written as } +3.678 \times 10^{-3}$$

$$-256.78 \text{ is written as } -2.5678 \times 10^{2}$$

Example

The form is

$$\text{sign} \times \text{mantissa} \times 10^{\text{exponent}}$$

or

$$\sigma \times m \times 10^{e}$$

Example: For

$$-2.5678 \times 10^{2}$$

$$\sigma = -1$$

$$m = 2.5678$$

$$e = 2$$

Floating Point Format for Binary Numbers

$$y = \sigma \times m \times 2^{e}$$

$$\sigma = \text{sign of number } (0 \text{ for } +\text{ve}, 1 \text{ for } -\text{ve})$$

$$m = \text{mantissa } [(1)_2 < m < (10)_2]$$

1 is not stored as it is always given to be 1.

$$e = \text{integer exponent}$$

Example

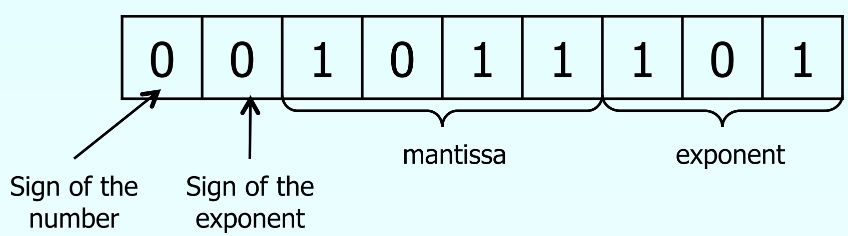
9 bit-hypothetical word

- the first bit is used for the sign of the number,
- the second bit for the sign of the exponent,
- the next four bits for the mantissa, and
- the next three bits for the exponent

$$(54.75)_{10} = (110110.11)_2 = (1.1011011)_2 \times 2^5$$

$$\cong (1.1011)_2 \times 2^{(101)_2}$$

We have the representation as



Machine Epsilon

Defined as the measure of accuracy and found by difference between 1 and the next number that can be represented

Example

Ten bit word

- Sign of number
- Sign of exponent
- Next four bits for exponent
- Next four bits for mantissa

0	0	0	0	0	0	0	0	0	0

$= (1)_{10}$

Next number →

0	0	0	0	0	0	0	0	0	1

$= (1.0001)_2 = (1.0625)_{10}$

$$\in_{mach} = 1.0625 - 1 = 2^{-4}$$

Relative Error and Machine Epsilon

The absolute relative true error in representing a number will be less then the machine epsilon

Example

$$(0.02832)_{10} \cong (1.1100)_2 \times 2^{-6}$$

$$= (1.1100)_2 \times 2^{-(0110)_2}$$

10 bit word (sign, sign of exponent, 4 for exponent, 4 for mantissa)

0	1	0	1	1	0	1	1	0	0

Sign of the number — Sign of the exponent — exponent — mantissa

$$(1.1100)_2 \times 2^{-(0110)_2} = 0.0274375$$

$$\in_a = \left| \frac{0.02832 - 0.0274375}{0.02832} \right|$$

$$= 0.034472 < 2^{-4} = 0.0625$$

IEEE-754 Format

32 bits for single precision

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sign — Biased Exponent — Mantissa

$$\text{Value} = (-1)^s \times (1.m)_2 \times 2^e$$

Exponent for 32 Bit IEEE-754

8 bits would represent

$$0 \le e' \le 255$$

Bias is 127; so subtract 127 from representation

$$-127 \le e \le 128$$

Actually

$$-126 \le e \le 127$$

0	0	0	0	0	0	0	1	is -126
1	1	1	1	1	1	1	0	is 127
0	0	0	0	0	0	0	0	for number zero
1	1	1	1	1	1	1	1	for infinity, NaN, etc.

IEEE-754 Format

The largest number by magnitude

$$(1.1........1)_2 \times 2^{127} = 3.40 \times 10^{+38}$$

The smallest number by magnitude

$$(1.00......0)_2 \times 2^{-126} = 2.18 \times 10^{-38}$$

Machine epsilon $= 2^{-23} = 1.19 \times 10^{-7}$

Sources of Error

Major: All Engineering Majors

Authors: Autar Kaw, Luke Snyder

http://numericalmethods.eng.usf.edu

Numerical Methods for STEM undergraduates

Two sources of numerical error

1) Round off error
2) Truncation error

Round off Error

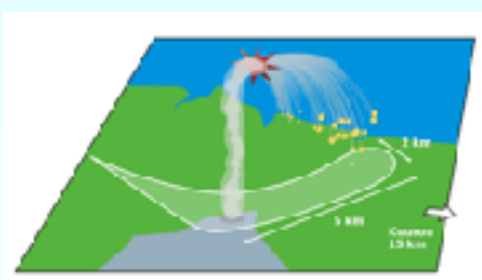
- Caused by representing a number approximately

$$\frac{1}{3} \cong 0.333333$$

$$\sqrt{2} \cong 1.4142...$$

Problems created by round off…

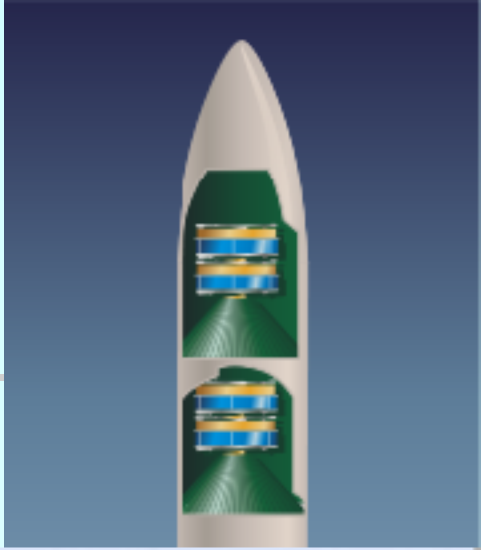
- **Ariane flight V88** was the failed maiden flight of the Arianespace Ariane 5 rocket, vehicle no. 501, on 4 June 1996. It carried the **Cluster** spacecraft, a constellation of four European Space Agency research satellites.
- inadequate protection against integer overflow led to an exception handled inappropriately



Ariane V first flight

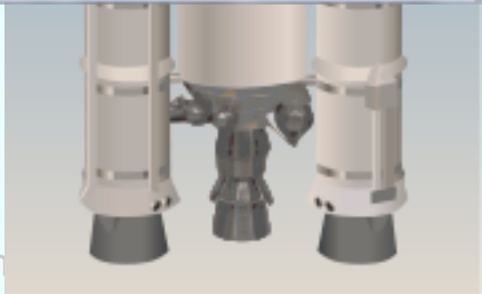
- data conversion from a 64-bit floating point number to a 16-bit signed integer value to overflow...

Problems created by round off…



```
        L_M_BV_32 := TDB.T_ENTIER_32S ((1.0/C_M_LSB_BV) *
                                        G_M_INFO_DERIVE(T_ALG.E_BV));
        if L_M_BV_32 > 32767 then
           P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
        elsif L_M_BV_32 < -32768 then
           P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
        else
           P_M_DERIVE(T_ALG.E_BV) := UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M
        end if;

501     P_M_DERIVE(T_ALG.E_BH) := UC_16S_EN_16NS (TDB.T_ENTIER_16S
                                        ((1.0/C_M_LSB_BH) *
                                        G_M_INFO_DERIVE(T_ALG.E_BH)))
     end LIRE_DERIVE;
```



Why visibility matters—the Ariane 5 crash

- Velocity was represented as a 64-bit float
- A conversion into a 16-bit signed integer caused an overflow
- The current velocity of Ariane 5 was too high to be represented as a 16-bit integer
- Error handling was suppressed for performance reasons

```
-- Vertical velocity bias as measured by sensor
L_M_BV_32 :=
   TBD.T_ENTIER_32S ((1.0/C_M_LSB_BV) *
   G_M_INFO_DERIVE(T_ALG.E_BV));
-- Check, if measured vertical velocity bias ban be
-- converted to a 16 bit int. If so, then convert
if L_M_BV_32 > 32767 then
   P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then
   P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
   P_M_DERIVE(T_ALG.E_BV) :=
            UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M_BV_32));
end if;
-- Horizontal velocity bias as measured by sensor
-- is converted to a 16 bit  int without checking
P_M_DERIVE(T_ALG.E_BH) :=
   UC_16S_EN_16NS (TDB.T_ENTIER_16S ((1.0/C_M_LSB_BH) *
   G_M_INFO_DERIVE(T_ALG.E_BH)));
```

*Source: http://moscova.inria.fr/~levy/talks/10enslongo/enslongo.pdf

Truncation error

- Error caused by truncating or approximating a mathematical procedure.

Example of Truncation Error

Taking only a few terms of a Maclaurin series to approximate e^x

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots\dots\dots\dots\dots\dots\dots$$

If only 3 terms are used,

$$Truncation\ \ Error = e^x - \left(1 + x + \frac{x^2}{2!}\right)$$

Another Example of Truncation Error

Using a finite Δx to approximate $f'(x)$

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

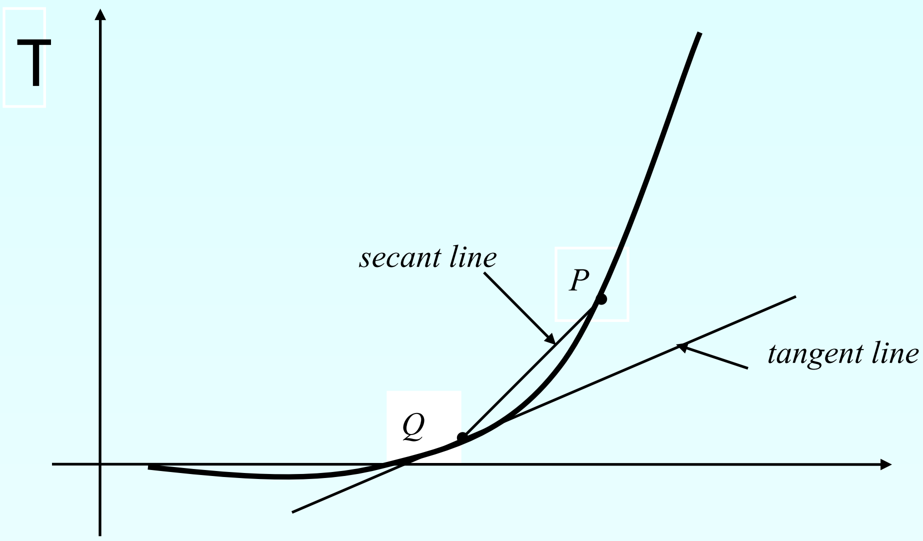
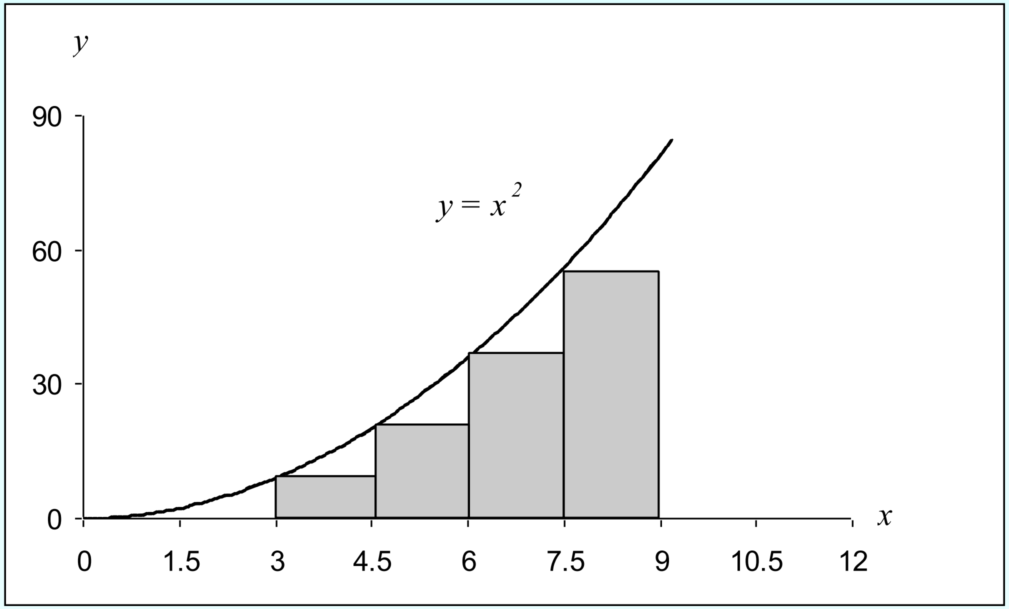


Figure 1. Approximate derivative using finite Δx

Another Example of Truncation Error

Using finite rectangles to approximate an integral.



Example 1 —Maclaurin series

Calculate the value of $e^{1.2}$ with an absolute relative approximate error of less than 1%.

$$e^{1.2} = 1 + 1.2 + \frac{1.2^2}{2!} + \frac{1.2^3}{3!} + \dots\dots\dots\dots\dots\dots$$

n	$e^{1.2}$	E_a	$\lvert\in_a\rvert\%$
1	1	—	—
2	2.2	1.2	54.545
3	2.92	0.72	24.658
4	3.208	0.288	8.9776
5	3.2944	0.0864	2.6226
6	3.3151	0.020736	0.62550

6 terms are required. How many are required to get at least 1 significant digit correct in your answer?

Example 2 —Differentiation

Find $f'(3)$ for $f(x) = x^2$ using $f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x}$

and $\Delta x = 0.2$

$$f'(3) = \frac{f(3+0.2) - f(3)}{0.2}$$

$$= \frac{f(3.2) - f(3)}{0.2} = \frac{3.2^2 - 3^2}{0.2} = \frac{10.24 - 9}{0.2} = \frac{1.24}{0.2} = 6.2$$

The actual value is

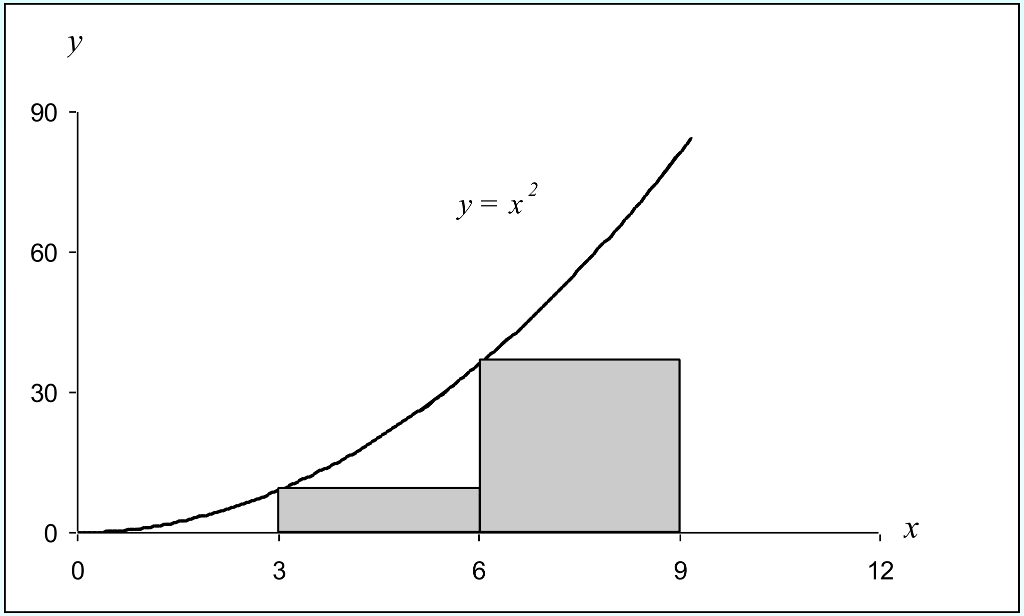
$$f'(x) = 2x, \quad f'(3) = 2 \times 3 = 6$$

Truncation error is then, $6 - 6.2 = -0.2$

Can you find the truncation error with $\Delta x = 0.1$

Example 3 — Integration

Use two rectangles of equal width to approximate the area under the curve for $f(x) = x^2$ over the interval $[3,9]$



$$\int_3^9 x^2 dx$$

Integration example (cont.)

Choosing a width of 3, we have

$$\int_3^9 x^2 dx = (x^2)\Big|_{x=3}(6-3) + (x^2)\Big|_{x=6}(9-6)$$

$$= (3^2)3 + (6^2)3$$

$$= 27 + 108 = 135$$

Actual value is given by

$$\int_3^9 x^2 dx = \left[\frac{x^3}{3}\right]_3^9 = \left[\frac{9^3 - 3^3}{3}\right] = 234$$

Truncation error is then

$$234 - 135 = 99$$

Can you find the truncation error with 4 rectangles?

Measuring Errors

Major: All Engineering Majors

Authors: Autar Kaw, Luke Snyder

http://numericalmethods.eng.usf.edu

Numerical Methods for STEM undergraduates

Why measure errors?

1) To determine the accuracy of numerical results.

2) To develop stopping criteria for iterative algorithms.

True Error

- Defined as the difference between the true value in a calculation and the approximate value found using a numerical method etc.

True Error = True Value – Approximate Value

Example—True Error

The derivative, $f'(x)$ of a function $f(x)$ can be approximated by the equation,

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

If $f(x) = 7e^{0.5x}$ and $h = 0.3$

a) Find the approximate value of $f'(2)$

b) True value of $f'(2)$

c) True error for part (a)

Example (cont.)

Solution:

a) For $x = 2$ and $h = 0.3$

$$f'(2) \approx \frac{f(2+0.3) - f(2)}{0.3}$$

$$= \frac{f(2.3) - f(2)}{0.3}$$

$$= \frac{7e^{0.5(2.3)} - 7e^{0.5(2)}}{0.3}$$

$$= \frac{22.107 - 19.028}{0.3} = 10.263$$

Example (cont.)

Solution:

b) The exact value of $f'(2)$ can be found by using our knowledge of differential calculus.

$$f(x) = 7e^{0.5x}$$

$$f'(x) = 7 \times 0.5 \times e^{0.5x}$$

$$= 3.5e^{0.5x}$$

So the true value of $f'(2)$ is

$$f'(2) = 3.5e^{0.5(2)}$$

$$= 9.5140$$

True error is calculated as

E_t = True Value – Approximate Value

$$= 9.5140 - 10.263 = -0.722$$

Relative True Error

- Defined as the ratio between the true error, and the true value.

$$\text{Relative True Error } (\in_t) = \frac{\text{True Error}}{\text{True Value}}$$

Example—Relative True Error

Following from the previous example for true error, find the relative true error for $f(x)=7e^{0.5x}$ at $f'(2)$ with $h=0.3$

From the previous example,

$$E_t = -0.722$$

Relative True Error is defined as

$$\in_t = \frac{\text{True Error}}{\text{True Value}}$$

$$= \frac{-0.722}{9.5140} = -0.075888$$

as a percentage,

$$\in_t = -0.075888 \times 100\% = -7.5888\%$$

Approximate Error

- What can be done if true values are not known or are very difficult to obtain?
- Approximate error is defined as the difference between the present approximation and the previous approximation.

Approximate Error (E_a) = Present Approximation – Previous Approximation

Example—Approximate Error

For $f(x)=7e^{0.5x}$ at $x=2$ find the following,

a) $f'(2)$ using $h=0.3$

b) $f'(2)$ using $h=0.15$

c) approximate error for the value of $f'(2)$ for part b)

Solution:

a) For $x=2$ and $h=0.3$

$$f'(x)\approx\frac{f(x+h)-f(x)}{h}$$

$$f'(2)\approx\frac{f(2+0.3)-f(2)}{0.3}$$

Example (cont.)

Solution: (cont.)

$$= \frac{f(2.3) - f(2)}{0.3}$$

$$= \frac{7e^{0.5(2.3)} - 7e^{0.5(2)}}{0.3}$$

$$= \frac{22.107 - 19.028}{0.3} = 10.263$$

b) For $x = 2$ and $h = 0.15$

$$f'(2) \approx \frac{f(2 + 0.15) - f(2)}{0.15}$$

$$= \frac{f(2.15) - f(2)}{0.15}$$

Example (cont.)

Solution: (cont.)

$$= \frac{7e^{0.5(2.15)} - 7e^{0.5(2)}}{0.15}$$

$$= \frac{20.50 - 19.028}{0.15} = 9.8800$$

c) So the approximate error, E_a is

E_a = Present Approximation – Previous Approximation

$$= 9.8800 - 10.263$$

$$= -0.38300$$

Relative Approximate Error

- Defined as the ratio between the approximate error and the present approximation.

$$\text{Relative Approximate Error } (\in_a) = \frac{\text{Approximate Error}}{\text{Present Approximation}}$$

Example—Relative Approximate Error

For $f(x) = 7e^{0.5x}$ at $x = 2$, find the relative approximate error using values from $h = 0.3$ and $h = 0.15$

Solution:

From Example 3, the approximate value of $f'(2) = 10.263$ using $h = 0.3$ and $f'(2) = 9.8800$ using $h = 0.15$

E_a = Present Approximation – Previous Approximation

$$= 9.8800 - 10.263$$

$$= -0.38300$$

Example (cont.)

Solution: (cont.)

$$\in_a = \frac{\text{Approximate Error}}{\text{Present Approximation}}$$

$$= \frac{-0.38300}{9.8800} = -0.038765$$

as a percentage,

$$\in_a = -0.038765 \times 100\% = -3.8765\%$$

Absolute relative approximate errors may also need to be calculated,

$$\left|\in_a\right| = \left|-0.038765\right| = 0.038765 \text{ or } 3.8765\%$$

How is Absolute Relative Error used as a stopping criterion?

If $|\in_a| \leq \in_s$ where $\in_s$ is a pre-specified tolerance, then no further iterations are necessary and the process is stopped.

If at least m significant digits are required to be correct in the final answer, then

$$|\in_a| \leq 0.5 \times 10^{2-m}$$

Table of Values

For $f(x) = 7e^{0.5x}$ at $x = 2$ with varying step size, h

h	$f'(2)$	$\lvert \in_a \rvert$	m
0.3	10.263	N/A	0
0.15	9.8800	0.038765%	3
0.10	9.7558	0.012731%	3
0.01	9.5378	0.024953%	3
0.001	9.5164	0.002248%	4

Propagation of Errors

Major: All Engineering Majors

Authors: Autar Kaw, Matthew Emmons

http://numericalmethods.eng.usf.edu

Numerical Methods for STEM undergraduates

Propagation of Errors

In numerical methods, the calculations are not made with exact numbers. How do these inaccuracies propagate through the calculations?

Example 1:

Find the bounds for the propagation in adding two numbers. For example if one is calculating X +Y where

$$X = 1.5 \pm 0.05$$

$$Y = 3.4 \pm 0.04$$

Solution

Maximum possible value of X = 1.55 and Y = 3.44

Maximum possible value of X + Y = 1.55 + 3.44 = 4.99

Minimum possible value of X = 1.45 and Y = 3.36.

Minimum possible value of X + Y = 1.45 + 3.36 = 4.81

Hence

$$4.81 \le X + Y \le 4.99.$$

Propagation of Errors In Formulas

If f is a function of several variables $X_1, X_2, X_3, \ldots\ldots, X_{n-1}, X_n$ then the maximum possible value of the error in f is

$$\Delta f \approx \left|\frac{\partial f}{\partial X_1}\Delta X_1\right| + \left|\frac{\partial f}{\partial X_2}\Delta X_2\right| + \ldots\ldots + \left|\frac{\partial f}{\partial X_{n-1}}\Delta X_{n-1}\right| + \left|\frac{\partial f}{\partial X_n}\Delta X_n\right|$$

Example 2:

The strain in an axial member of a square cross-section is given by

$$\in = \frac{F}{h^2 E}$$

Given

$$F = 72 \pm 0.9 \text{ N}$$
$$h = 4 \pm 0.1 \text{ mm}$$
$$E = 70 \pm 1.5 \text{ GPa}$$

Find the maximum possible error in the measured strain.

Example 2:

Solution

$$\in = \frac{72}{(4\times10^{-3})^2(70\times10^9)}$$

$$= 64.286\times10^{-6}$$

$$= 64.286\mu$$

$$\Delta\in = \left|\frac{\partial\in}{\partial F}\Delta F\right| + \left|\frac{\partial\in}{\partial h}\Delta h\right| + \left|\frac{\partial\in}{\partial E}\Delta E\right|$$

Example 2:

$$\frac{\partial \in}{\partial F} = \frac{1}{h^2 E} \qquad \frac{\partial \in}{\partial h} = -\frac{2F}{h^3 E} \qquad \frac{\partial \in}{\partial E} = -\frac{F}{h^2 E^2}$$

Thus

$$\Delta E = \left|\frac{1}{h^2 E}\Delta F\right| + \left|\frac{2F}{h^3 E}\Delta h\right| + \left|\frac{F}{h^2 E^2}\Delta E\right|$$

$$= \left|\frac{1}{(4\times 10^{-3})^2(70\times 10^9)}\times 0.9\right| + \left|\frac{2\times 72}{(4\times 10^{-3})^3(70\times 10^9)}\times 0.0001\right|$$

$$+ \left|\frac{72}{(4\times 10^{-3})^2(70\times 10^9)^2}\times 1.5\times 10^9\right|$$

$$= 5.3955\mu$$

Hence

$$\in = (64.286\mu \pm 5.3955\mu)$$

Example 3:

Subtraction of numbers that are nearly equal can create unwanted inaccuracies. Using the formula for error propagation, show that this is true.

Solution

Let

$$z = x - y$$

Then

$$|\Delta z| = \left|\frac{\partial z}{\partial x}\Delta x\right| + \left|\frac{\partial z}{\partial y}\Delta y\right|$$

$$= |(1)\Delta x| + |(-1)\Delta y|$$

$$= |\Delta x| + |\Delta y|$$

So the relative change is

$$\left|\frac{\Delta z}{z}\right| = \frac{|\Delta x| + |\Delta y|}{|x - y|}$$

Example 3:

For example if

$$x = 2 \pm 0.001$$

$$y = 2.003 \pm 0.001$$

$$\left|\frac{\Delta z}{z}\right| = \frac{|0.001| + |0.001|}{|\,2 - 2.003\,|}$$

$$= 0.6667$$

$$= 66.67\%$$

Taylor Series Revisited

Major: All Engineering Majors

Authors: Autar Kaw, Luke Snyder

http://numericalmethods.eng.usf.edu

Numerical Methods for STEM undergraduates

What is a Taylor series?

Some examples of Taylor series which you must have seen

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots$$

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

General Taylor Series

The general form of the Taylor series is given by

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \frac{f'''(x)}{3!}h^3 + \cdots$$

provided that all derivatives of f(x) are continuous and exist in the interval [x,x+h]

What does this mean in plain English?

As Archimedes would have said, “Give me the value of the function at a single point, and the value of all (first, second, and so on) its derivatives at that single point, and I can give you the value of the function at any other point” (fine print excluded)

Example—Taylor Series

Find the value of $f(6)$ given that $f(4)=125$, $f'(4)=74$, $f''(4)=30$, $f'''(4)=6$ and all other higher order derivatives of $f(x)$ at $x=4$ are zero.

Solution:

$$f(x+h)=f(x)+f'(x)h+f''(x)\frac{h^2}{2!}+f'''(x)\frac{h^3}{3!}+\cdots$$

$$x=4$$

$$h=6-4=2$$

Example (cont.)

Solution: (cont.)

Since the higher order derivatives are zero,

$$f(4+2) = f(4) + f'(4)2 + f''(4)\frac{2^2}{2!} + f'''(4)\frac{2^3}{3!}$$

$$f(6) = 125 + 74(2) + 30\left(\frac{2^2}{2!}\right) + 6\left(\frac{2^3}{3!}\right)$$

$$= 125 + 148 + 60 + 8$$

$$= 341$$

Note that to find $f(6)$ exactly, we only need the value of the function and all its derivatives at some other point, in this case $x = 4$

Derivation for Maclaurin Series for e^x

Derive the Maclaurin series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

The Maclaurin series is simply the Taylor series about the point x=0

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2!} + f'''(x)\frac{h^3}{3!} + f''''(x)\frac{h^4}{4} + f'''''(x)\frac{h^5}{5} + \cdots$$

$$f(0+h) = f(0) + f'(0)h + f''(0)\frac{h^2}{2!} + f'''(0)\frac{h^3}{3!} + f''''(0)\frac{h^4}{4} + f'''''(0)\frac{h^5}{5} + \cdots$$

Derivation (cont.)

Since $f(x) = e^x$, $f'(x) = e^x$, $f''(x) = e^x$, ..., $f^n(x) = e^x$ and $f^n(0) = e^0 = 1$

the Maclaurin series is then

$$f(h) = (e^0) + (e^0)h + \frac{(e^0)}{2!}h^2 + \frac{(e^0)}{3!}h^3 \ldots$$

$$= 1 + h + \frac{1}{2!}h^2 + \frac{1}{3!}h^3 \ldots$$

So,

$$f(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

Error in Taylor Series

The Taylor polynomial of order n of a function f(x) with (n+1) continuous derivatives in the domain [x,x+h] is given by

$$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2!} + \cdots + f^{(n)}(x)\frac{h^n}{n!} + R_n(x)$$

where the remainder is given by

$$R_n(x) = \frac{(x-h)^{n+1}}{(n+1)!} f^{(n+1)}(c)$$

where

$$x < c < x + h$$

that is, c is some point in the domain [x,x+h]

Example—error in Taylor series

The Taylor series for e^x at point $x = 0$ is given by

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots$$

It can be seen that as the number of terms used increases, the error bound decreases and hence a better estimate of the function can be found.

How many terms would it require to get an approximation of e^1 within a magnitude of true error of less than 10^{-6}.

Example—(cont.)

Solution:

Using $(n+1)$ terms of Taylor series gives error bound of

$$R_n(x) = \frac{(x-h)^{n+1}}{(n+1)!} f^{(n+1)}(c) \qquad x = 0, h = 1, f(x) = e^x$$

$$R_n(0) = \frac{(0-1)^{n+1}}{(n+1)!} f^{(n+1)}(c)$$

$$= \frac{(-1)^{n+1}}{(n+1)!} e^c$$

Since

$$x < c < x + h$$

$$0 < c < 0 + 1$$

$$0 < c < 1$$

$$\frac{1}{(n+1)!} < |R_n(0)| < \frac{e}{(n+1)!}$$

Example—(cont.)

Solution: (cont.)

So if we want to find out how many terms it would require to get an approximation of e^1 within a magnitude of true error of less than 10^{-6},

$$\frac{e}{(n+1)!} < 10^{-6}$$

$$(n+1)! > 10^6 e$$

$$(n+1)! > 10^6 \times 3$$

$$n \geq 9$$

So 9 terms or more are needed to get a true error less than 10^{-6}

- Make yourself acquainted with your computer desktop (Unix environment). Use the Unix commands **ls**, **df**, **ps**, test the use of an editor of your choice to write small programs or texts (e.g. **vi, emacs, joe, nano,** ...).
- Check how you can produce plots, e.g. using the **gnuplot** program, **matplotlib** or any other software of your choice.

Your code should be in a programming language of your choice (support can only be offered for Python, Fortran or C, C++).
Ensure readable and organized code:

- using naming conventions for variables;
- placing whitespaces, indentations and tabs within code;
- adding comments throughout to aid in interpretation.

Tv

- **Exercise 1, 6 points**: Round-off Errors
  Convert the decimal number $(-0.004831)_{10}$ into a binary format used for the hypothetical ten-bit word presented in the lecture. Compute the true error and the relative true error (absolute values) made by the ten-bit representation of $(-0.004831)_{10}$. (No programming necessary.)

- **Exercise 2, 6 points**: Truncation Errors
  Calculate the value of $e^{1.5}$ using the Taylor series of e^x. Increase the number of terms used in the Taylor series until the relative approximate error (absolute value) is less than 0.1 %. Document the results in a table, the code in a printout. Do this for at least two different machine precisions, e.g. Python: `numpy.float32`, `numpy.float64`, `numpy.single`, `numpy.double` or C++: `float a; double d;` for comparison.

- **Exercise 3, 8 points**: Machine ε
  Solve the quadratic equation $x^2 + x + c = 0$ directly using the quadrature $x_1 = (-1 + \sqrt{1 - 4c})/2$, for $0 \leq c \leq 1/4$. Prepare a computer program, which outputs x_1 as a function of c. What is the smallest c which produces a correct solution for $x1 \neq 0$? Hint $c_{init} = 0.25$ then $c_{new} \leftarrow c_{old} \times 0.5$. Does $\times 0.9$ make a difference? Relate this to the machine ε for single precision. How can you obtain a more reliable result even numerically for small c by rewriting the quadrature expression? Please print this for two different machine precisions as well.

- Questions?