Introduction to Numerical Methods for Solving Partial Differential Equations

Benson Muite

benson.muite@ut.ee
http://kodu.ut.ee/~benson
http://en.wikibooks.org/wiki/Parallel_Spectral_Numerical_Methods

25 March 2015
Outline

• Review
• Aim
• The heat equation
• Methods for solving the heat equation
• Fourier Series
• The Allen Cahn Equation
• Summary
• References
Ordinary Differential Equations: Numerical Schemes

- **Forward Euler method**
  \[
  \frac{y^{n+1} - y^n}{\delta t} = f(y^n)
  \]

- **Backward Euler method**
  \[
  \frac{y^{n+1} - y^n}{\delta t} = f(y^{n+1})
  \]

- **Implicit Midpoint rule**
  \[
  \frac{y^{n+1} - y^n}{\delta t} = f\left(\frac{y^{n+1} + y^n}{2}\right)
  \]

- **Crank Nicolson Method**
  \[
  \frac{y^{n+1} - y^n}{\delta t} = \frac{f(y^{n+1}) + f(y^n)}{2}
  \]

- **Other Methods**: Runge Kutta, Adams Bashforth, Backward differentiation, splitting
The Heat Equation

- Used to model diffusion of heat, species,

- 1D

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}
\]

- 2D

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}
\]

- 3D

\[
\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}
\]

- Not always a good model, since it has infinite speed of propagation

- Strong coupling of all points in domain make it computationally intensive to solve in parallel
Linear Schrödinger Equation

- Used to model quantum mechanical phenomena and often appears in simplified wave propagation models
- 1D
  \[ i \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \]
- 2D
  \[ i \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \]
- 3D
  \[ i \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \]
- Looks like a heat equation with imaginary time
- Strong coupling of all points in domain make it computationally intensive to solve in parallel
The Wave Equation

- Used to model propagation of sound, light
- 1D
  \[
  \frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2}
  \]
- 2D
  \[
  \frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}
  \]
- 3D
  \[
  \frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}
  \]
- Has finite speed of propagation
- Finite signal propagation speed sometimes useful in parallelization since there is no coupling between grid points that are far apart, hence smaller communication requirements
- If propagation speed is very fast, communication requirements still important
The Poisson Equation

- Used to model static deflection of a drumhead

1D

\[ f(x) = \frac{\partial^2 u}{\partial x^2} \]

2D

\[ f(x, y) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \]

3D

\[ f(x, y, z) = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \]

- No time dependence, but also arises in time discretizations of time dependent partial differential equations
Burgers’ Equation

- Simple model for gas dynamics, also traffic
- Inviscid form (can develop shocks)
  \[ u_t = -uu_x \]
- Viscous form
  \[ u_t = -uu_x + \epsilon u_{xx} \]
Navier-Stokes Equations

- Models motion of fluids
- Consider incompressible case only (low speed motion relative to speed of sound)

\[
\rho \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \Delta \mathbf{u}
\]
\[
\nabla \cdot \mathbf{u} = 0.
\]

- \( p \) pressure, \( \mu \) viscosity, \( \rho \), density
- 2D \( \mathbf{u}(x, y) = (u(x, y), v(x, y)) \)
- 3D \( \mathbf{u}(x, y, z) = (u(x, y, z), v(x, y, z), w(x, y, z)) \)
Maxwell's Equations

- Model propagation of electromagnetic waves

\[ \mathbf{D}_t - \nabla \times \mathbf{H} = 0 \]

\[ \mathbf{B}_t + \nabla \times \mathbf{E} = 0 \]

- with the relation between the electromagnetic components given by the constitutive relations:

\[ \mathbf{D} = \varepsilon_o(x, y, z)\mathbf{E} \]

\[ \mathbf{B} = \mu_o(x, y, z)\mathbf{H} \]
Summary on Models

• Many diverse models
• In 2D and 3D, parallel computing is very useful for getting numerical solutions in reasonable time
• Typical applications are in physics, chemistry, biology and engineering
• Related models also appear in social sciences, though usefulness for predictability of real world data is less clear
Aim to simulate this,


- Visualization around Formula 1 Race Car by Renato N. Elias, Rio de Janeiro, Brazil
or this


- Visualization around Formula 1 Race Car by Renato N. Elias, Rio de Janeiro, Brazil
Maybe even this

- https://youtu.be/FT1J9l9KTkw
- https://youtu.be/MsDgw8_cb90
- https://youtu.be/Su8qzC4HHVs
A simple model: The Heat Equation
The Heat Equation

- How does the temperature of the rod vary with time?
The Heat Equation: Model 1

- Middle of rod is initially hot due to previous heating (e.g., oven in house that has just been turned off)
- Temperature on the two sides is 0 (winter and cold outside the house)
- Assume discrete uniformly space time, and discrete space with molecules at each coordinate point
- Assume temperature in bar is directly proportional to heat, with the same proportionality constant throughout the bar
- At next time step, a molecule will
  - with probability 0.5 not transfer its current heat
  - with probability 0.25 transfer half of its heat to its left neighbor
  - with probability 0.25 transfer half of its heat to its right neighbor
The Heat Equation: Model 1

• Let $T_i^n$ denote the temperature at position $i$ at time $n$. Particles at the end have a fixed temperature/heat they can transfer, but they always remain at the same temperature.

$$T_i^{n+1} =$$

Not transfer heat

$$0.0 T_{i-1}^n + T_i^n + 0.5 T_{i+1}^n \quad \text{probability } 0.75 \times 0.5 \times 0.25$$
$$0.5 T_{i-1}^n + T_i^n + 0.0 T_{i+1}^n \quad \text{probability } 0.25 \times 0.5 \times 0.75$$
$$0.5 T_{i-1}^n + T_i^n + 0.5 T_{i+1}^n \quad \text{probability } 0.25 \times 0.5 \times 0.25$$
$$0.0 T_{i-1}^n + T_i^n + 0.0 T_{i+1}^n \quad \text{probability } 0.75 \times 0.5 \times 0.75$$

Transfer heat left

$$0.0 T_{i-1}^n + 0.5 T_i^n + 0.5 T_{i+1}^n \quad \text{probability } 0.75 \times 0.25 \times 0.25$$
$$0.5 T_{i-1}^n + 0.5 T_i^n + 0.0 T_{i+1}^n \quad \text{probability } 0.25 \times 0.25 \times 0.75$$
$$0.5 T_{i-1}^n + 0.5 T_i^n + 0.5 T_{i+1}^n \quad \text{probability } 0.25 \times 0.25 \times 0.25$$
$$0.0 T_{i-1}^n + 0.5 T_i^n + 0.0 T_{i+1}^n \quad \text{probability } 0.75 \times 0.25 \times 0.75$$

Transfer heat right

$$0.0 T_{i-1}^n + 0.5 T_i^n + 0.5 T_{i+1}^n \quad \text{probability } 0.75 \times 0.25 \times 0.25$$
$$0.5 T_{i-1}^n + 0.5 T_i^n + 0.0 T_{i+1}^n \quad \text{probability } 0.25 \times 0.25 \times 0.75$$
$$0.5 T_{i-1}^n + 0.5 T_i^n + 0.5 T_{i+1}^n \quad \text{probability } 0.25 \times 0.25 \times 0.25$$
$$0.0 T_{i-1}^n + 0.5 T_i^n + 0.0 T_{i+1}^n \quad \text{probability } 0.75 \times 0.25 \times 0.75$$
The Heat Equation: Model 1

- DEMO
The Heat Equation: Model 2

- Probability is hard!
- It takes a long time
- May only interested in average temperature/heat, not instantaneous fluctuations
- Take averages

\[ T_i^{n+1} = (0.25 \times 0.5) T_{i-1}^n + (0.5 + 0.25 \times 0.5 + 0.25 \times 0.5) T_i^n + (0.25 \times 0.5) T_{i+1}^n \]

\[ T_i^{n+1} = 0.125 T_{i-1}^n + 0.75 T_i^n + 0.125 T_{i+1}^n \]

or

\[ T_i^{n+1} = T_i^n + \frac{T_{i-1}^n - 2T_i^n + T_{i+1}^n}{8} \]

- The last line looks like a difference approximation for a differential equation
The Heat Equation: Model 3

- Let us find a differential equation!
- Make the time increment small

\[ T_i^{n+1} = T_i^n + \frac{T_{i-1}^n - 2T_i^n + T_{i+1}^n}{8} \]

\[ \frac{T_i^{n+1} - T_i^n}{\delta t} = \frac{T_{i-1}^n - 2T_i^n + T_{i+1}^n}{8\delta t} \]
The Heat Equation: Model 3

- Let us find a differential equation!
- Make the space increment small

\[
\frac{T_i^{n+1} - T_i^n}{\delta t} = \frac{T_{i-1}^n - 2T_i^n + T_{i+1}^n}{8\delta t}
\]

\[
\frac{T_i^{n+1} - T_i^n}{\delta t} = \frac{(\delta x)^2}{8\delta t} \left( \frac{T_{i-1}^n - T_i^n}{\delta x} - \frac{T_i^n - T_{i+1}^n}{\delta x} \right)
\]

- Let \( \delta x \rightarrow 0 \) and \( \delta t \rightarrow 0 \) such that \((\delta x)^2 = \delta t\) we get

\[
\frac{\partial T}{\partial t} = \frac{1}{8} \frac{\partial^2 T}{\partial x^2}
\]

- Constant in continuum formulation depends on physics and is usually measured experimentally, or determined from microscopic (many particle) simulations.
The Heat Equation: Numerical Solution Methods

- Probabalistic
- Finite Difference
- Finite Volume
- Finite Element
- Spectral
Finite Difference Method

- Approximate derivatives by difference quotients
- Simple method to derive and implement
- Convergence rates tend not to be great
- Difficult to use for complicated geometries
- Tends to scale well since communication requirements are low
Finite Difference Method for Heat Equation

- \( u_t = 8u_{xx} \)
- Using backward Euler time stepping:
  \[
  \frac{u_{i}^{n+1} - u_{i}^{n}}{\delta t} = 8 \frac{u_{i-1}^{n+1} - 2u_{i}^{n+1} + u_{i+1}^{n+1}}{(\delta x)^2}
  \]
- Using forward Euler time stepping (strong stability restrictions):
  \[
  \frac{u_{i}^{n+1} - u_{i}^{n}}{\delta t} = 8 \frac{u_{i-1}^{n} - 2u_{i}^{n} + u_{i+1}^{n}}{(\delta x)^2}
  \]
Finite Difference Method for Heat Equation

- Simple method to derive and implement
- Hardest part for implicit schemes is solution of resulting linear system of equations
- Explicit schemes typically have stability restrictions or can always be unstable
- Convergence rates tend not to be great – to get an accurate solution, a large number of grid points are needed
- Difficult to use for complicated geometries
- Tends to scale well since communication requirements are low
Finite Volume Method for Heat Equation

- $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$ or $\frac{\partial u}{\partial x} = v$ and $\frac{\partial u}{\partial t} = \frac{\partial v}{\partial x}$
- Consider a cell averaged integral, then use implicit midpoint rule

\[
\int_{x_i}^{x_{i+1}} u_x dx = \int_{x_i}^{x_{i+1}} v dx
\]
\[
u_{i+1} - u_{i} = \int_{x_i}^{x_{i+1}} v dx
\]
\[
\frac{u_{i+1}^{n+1} + u_{i+1}^{n} - u_{i+1}^{n} - u_{i}^{n}}{2} = \frac{v_{i+1}^{n+1} + v_{i+1}^{n} + v_{i}^{n+1} + v_{i}^{n}}{2} \delta x
\]

\[
\int_{x_i}^{x_{i+1}} u_t dx = \int_{x_i}^{x_{i+1}} v_x dx
\]
\[
\int_{x_i}^{x_{i+1}} u_t dx = v_{i+1} - v_{i}
\]
\[
\frac{u_{i+1}^{n+1} + u_{i+1}^{n} - u_{i+1}^{n} - u_{i}^{n}}{2 \delta t} \delta x = \frac{v_{i+1}^{n+1} + v_{i+1}^{n} - v_{i}^{n+1} - v_{i}^{n}}{2}
\]

- Several ways of approximating the integrals. The one above is a little unusual, most finite volume schemes use left sided or right sided approximations.
Finite Volume Method for Heat Equation

- For implicit schemes, hardest part is solving the system of equations that results
- Explicit schemes parallelize very well, however a large number of grid points are usually needed to get accurate results
- Automated construction of simple finite volume schemes is possible, making them popular in packages
- No convergence theory for high order finite volume schemes
- Tricky to do complicated geometries accurately
Finite Element Method for Heat Equation

- \( \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \)
- Assume \( u(x, t) \approx \tilde{u}(x, t) = \sum_i \phi_i(x) T_i(t) \), where \( \phi_i(x) \) is zero for \( x > x_{i+1} \) and \( x < x_{i-1} \).
- A simple choice is \( \psi \) is the triangle hat function, \( x_{i+1} - x \) for \( x \in [x_i, x_{i+1}] \) and \( x - x_{i-1} \) for \( x \in [x_{i-1}, x_i] \)
- Now need to find \( T_i(t) \), which will be evaluated using finite differences.
Consider multiplying the heat equation by a polynomial, $\phi_j$ that is only non-zero over a few grid points, then integrate by parts,

$$
\int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial t} \phi_i \, dx = \int_{x_{i-1}}^{x_{i+1}} \frac{\partial^2 \tilde{u}}{\partial x^2} \phi_i \, dx
$$

$$
\int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial t} \phi_i \, dx = - \int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial x} \frac{\partial \phi_i}{\partial x} \, dx
$$
Finite Element Method for Heat Equation

• Then use implicit midpoint rule

\[
\int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial t} \phi_i \, dx = - \int_{x_{i-1}}^{x_{i+1}} \frac{\partial \tilde{u}}{\partial x} \frac{\partial \phi_i}{\partial x} \, dx
\]

\[
\int_{x_{i-1}}^{x_{i+1}} \frac{\tilde{u}^{n+1} - \tilde{u}^n}{\delta t} \phi_i \, dx = - \int_{x_{i-1}}^{x_{i+1}} \frac{1}{2} \left( \frac{\partial \tilde{u}^n}{\partial x} + \frac{\partial \tilde{u}^{n+1}}{\partial x} \right) \frac{\partial \phi_i}{\partial x} \, dx
\]

\[
\frac{1}{2} \left[ \frac{\tilde{u}^{n+1} - \tilde{u}_i}{\delta t} \phi_i \bigg|_{x_{i+1}} + \frac{\tilde{u}^{n+1}_i - \tilde{u}^n_i}{\delta t} \phi_i \bigg|_{x_{i-1}} \right] 2 \delta x
\]

\[
= - \frac{1}{2} \left[ \frac{1}{2} \left( \frac{\partial \tilde{u}^{n+1}}{\partial x} \bigg|_{x_{i+1}} + \frac{\partial \tilde{u}^n}{\partial x} \bigg|_{x_{i+1}} \right) \frac{\partial \phi_i}{\partial x} \bigg|_{x_{i+1}} \right] 2 \delta x
\]

\[
- \frac{1}{2} \left[ \frac{1}{2} \left( \frac{\partial \tilde{u}^{n+1}}{\partial x} \bigg|_{x_{i-1}} + \frac{\partial \tilde{u}^n}{\partial x} \bigg|_{x_{i-1}} \right) \frac{\partial \phi_i}{\partial x} \bigg|_{x_{i-1}} \right] 2 \delta x
\]

• Since \( \phi \) are known before hand, can re-write this as matrix vector products, so need to solve a linear system at each time step.
Finite Element Method for Heat Equation

- Several other ways of approximating the integrals, can extend to multiple dimensions.
- Weak formulation allows for solution of equations where second derivative is not naturally defined.
- Large mathematical community developing convergence theory for these methods.
- Well suited to complicated geometries.
- Rather difficult to implement compared to other schemes because of integrals that need to be computed.
- Used in many codes, but typically codes are hand written to obtain high efficiency.
- For implicit time discretizations, solving the linear system of equations that results can be most time consuming part.
Fourier Spectral Method for Heat Equation
Fourier Series: Separation of Variables 1

\[
\frac{dy}{dt} = y
\]

\[
\frac{dy}{y} = dt
\]

\[
\int \frac{dy}{y} = \int dt
\]

\[
\ln y + a = t + b
\]

\[
e^{\ln y + a} = e^{t + b}
\]

\[
e^{\ln y} e^a = e^t e^b
\]

\[
y = \frac{e^b}{e^a} e^t
\]

\[
y(t) = ce^t
\]
Fourier Series: Separation of Variables 2

- \[ \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \]

- Suppose \( u = X(x)T(t) \)

- \[ \frac{dT}{dt}(t) = \frac{d^2X}{dx^2}(x) \quad \Rightarrow \quad -C, \]

- Solving each of these separately and then using linearity we get a general solution

\[ \sum_{n=0}^{\infty} \alpha_n \exp(-C_n t) \sin(\sqrt{C_n}x) + \beta_n \exp(-C_n t) \cos(\sqrt{C_n}x) \]
Fourier Series: Separation of Variables

- How do we find a particular solution?
- Suppose \( u(x, t = 0) = f(x) \)
- Suppose \( u(0, t) = u(2\pi, t) \) and \( u_x(0, t) = u_x(2\pi, t) \) then recall

\[
\begin{align*}
\int_0^{2\pi} \sin(nx) \sin(mx) &= \begin{cases} 
\pi & m = n \\
0 & m \neq n
\end{cases}, \\
\int_0^{2\pi} \cos(nx) \cos(mx) &= \begin{cases} 
\pi & m = n \\
0 & m \neq n
\end{cases}, \\
\int_0^{2\pi} \cos(nx) \sin(mx) &= 0.
\end{align*}
\]
Fourier Series: Separation of Variables 4

• So if

\[ f(x) = \sum_{n=0}^{\infty} \alpha_n \sin(nx) + \beta_n \cos(nx). \]

• then

\[ \alpha_n = \frac{\int_0^{2\pi} f(x) \sin(nx) \, dx}{\int_0^{2\pi} \sin^2(nx) \, dx} \]
\[ \beta_n = \frac{\int_0^{2\pi} f(x) \cos(nx) \, dx}{\int_0^{2\pi} \cos^2(nx) \, dx}. \]

• and

\[ u(x, t) = \sum_{n=0}^{\infty} \exp(-n^2 t) [\alpha_n \sin(nx) + \beta_n \cos(nx)]. \]
• The Fast Fourier Transform allows one to find good approximations to $\alpha_n$ and $\beta_n$ when the solution is found at a finite number of evenly spaced grid points.
• By rescaling, can consider intervals other than $[0, 2\pi)$.
• Fourier transform also works on infinite intervals, but require function to decay to the same constant value at $\pm\infty$. 
Complex Fourier Series

- By using Euler’s formula, one can get a simpler expression for a Fourier series where sine and cosine are combined.

\[ u = \sum_{n=-\infty}^{\infty} \gamma_n \exp(inx) \quad x \in [0, 2\pi) \]

Source: http://en.wikipedia.org/wiki/File:Euler%27s_formula.svg
A Computational Algorithm for Computing An Approximate Fourier Transform

- Analytic method of computing Fourier transform can be tedious
- Can use quadrature to numerically evaluate Fourier transforms – $O(n^2)$ operations
- Gauss and then Cooley and Tukey found $O(n \log n)$ algorithm
- Key observation is to use factorization and recursion
- Modern computers use variants of this idea that are more suitable for computer hardware where moving data is more expensive than floating point operations
Example pseudo code to compute a radix 2 out of place DFT where x has length that is a power of 2

1: procedure X_0,\ldots,N-1 (ditfft2(x,N,s))
2: \quad DFT of (x_0, x_s, x_{2s}, \ldots, x_{(N-1)s})
3: \quad if N=1 then
4: \quad \quad trivial size-1 DFT of base case
5: \quad \quad X_0 \leftarrow X_0
6: \quad else
7: \quad \quad DFT of (x_0, x_{2s}, x_{4s}, \ldots)
8: \quad \quad X_0,\ldots,N/2-1 \leftarrow ditfft2(x, N/2, 2s)
9: \quad \quad DFT of (x_s, x_{s+2s}, x_{s+4s}, \ldots)
10: \quad \quad X_{N/2},\ldots,N-1 \leftarrow ditfft2(x+s, N/2, 2s)
11: \quad Combine DFTs of two halves into full DFT
12: \quad for k = 0 \rightarrow N/2 - 1 do
13: \quad \quad t \leftarrow X_k
14: \quad \quad X_k \leftarrow t + \exp(-2\pi i k / N)X_{k+N/2}
15: \quad \quad X_{k+N/2} \leftarrow t - \exp(-2\pi i k / N)X_{k+N/2}
16: \quad end for
17: end if
18: end procedure

The Heat Equation: Finding Derivatives and Timestepping

- Let
  \[ u(x) = \sum_k \hat{u}_k \exp(ikx) \]

- then
  \[ \frac{d^\nu u}{dx^\nu} = \sum (ik)^\nu \hat{u}_k \exp(ikx). \]

- Consider \( u_t = u_{xx} \), which is approximated by
  \[
  \frac{\partial \hat{u}_k}{\partial t} = (ik)^2 \hat{u}_k \\
  \frac{\hat{u}_k^{n+1} - \hat{u}_k^n}{\delta t} = (ik)^2 \hat{u}_k^{n+1} \\
  \hat{u}_k^{n+1} [1 - \delta t (ik)^2] = \hat{u}_k^n \\
  \hat{u}_k^{n+1} = \frac{\hat{u}_k^n}{[1 - \delta t (ik)^2]}.
  \]
The Heat Equation: Finding Derivatives and Timestepping

- Python demonstration
The Allen Cahn Equation: Implicit-Explicit Method

- Consider $u_t = \epsilon u_{xx} + u - u^3$, which is approximated by backward Euler for the linear terms and forward Euler for the nonlinear terms.

\[
\frac{\partial \hat{u}}{\partial t} = \epsilon (ik)^2 \hat{u} + \hat{u} - \hat{u}^3
\]

\[
\frac{\hat{u}^{n+1} - \hat{u}^n}{\delta t} = \epsilon (ik)^2 u^{n+1} + \hat{u}^n - (u^n)^3
\]
The Allen Cahn Equation: Implicit-Explicit Method

- Python demonstration
Key Concepts

- Differential equations used to model physical phenomena - approximate model
- Microscopic model can suggest ways to find approximate solutions on computers
- Several methods introduced, probabilistic, finite difference, finite element, spectral (Fourier transform)
- Need to check simulations for discretization errors, then before using for prediction, understand modeling errors
- Time stepping, Forward Euler, Backward Euler, Implicit Midpoint Rule, Implicit-Explicit
References

- Corral M. Vector Calculus http://www.mecmath.net/
References

• McGrattan, K. et al. Fire Dynamics Simulator
  http://www.nist.gov/el/fire_research/fds_smokeview.cfm
• Quarteroni, A. Sacco, R. and Saleri, F. “Numerical Mathematics”
• Sauer T. Numerical Analysis Pearson (2012)
• Süli, E. “Finite element methods for partial differential equations”
  people.maths.ox.ac.uk/suli/fem.pdf
• Trefethen, L.N. “Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations”
  http://people.maths.ox.ac.uk/trefethen/pdetext.html
• Vainikko E. “Scientific Computing Lectures”
  https://courses.cs.ut.ee/2013/scicomp/fall/Main/Lectures
• Vainikko E. Fortran 95 Ja MPI Tartu Ülikool Kirjastus (2004)
  http://kodu.ut.ee/~eero/PC/F95jaMPI.pdf
Acknowledgements

- Oleg Batrașev
- Leonid Dorogin
- Michael Quell
- Eero Vainikko
- The Blue Waters Undergraduate Petascale Education Program administered by the Shodor foundation
- The Division of Literature, Sciences and Arts at the University of Michigan
Extra Slides
Fun with Fourier Series

- A Fourier series can represent a regular k-gon in the complex plane
- \[ f(t) = \sum_{n=-\infty}^{+\infty} \frac{1}{(1 + kn)^2} \exp[i(1 + kn)t] \quad t \in (-\pi, \pi) \]


Monte Carlo Method: A Probabilistic Way to Calculate Integrals

- Recall

\[ \bar{f} = \frac{1}{b - a} \int_a^b f(x) \, dx \]

- Hence given \( \bar{f} \)

\[ \int_a^b f(x) \, dx = (b - a) \bar{f} \]

- Doing the same in 2 dimensions and estimating the error using the standard deviation

\[ \int \int_R f(x, y) \, dA \approx A(R) \bar{f} \pm A(R) \sqrt{\frac{\bar{f}^2 - (\bar{f})^2}{N}} \]

- Approximate \( \bar{f} \) by random sampling

\[ \bar{f} \approx \frac{\sum_{i=1}^{N} f(x_i, y_i)}{N} \quad \text{and} \quad \bar{f}^2 \approx \frac{\sum_{i=1}^{N} (f(x_i, y_i))^2}{N} \]
Monte Carlo Method: Python Program

"""
A program to approximate an integral using a Monte Carlo method

This could be made faster by using vectorization, however it is kept as simple as possible for clarity and ease of translation into other languages

"""

import math
import numpy
import time
numpoints=65536  # number of random sample points
I2d=0.0          # initialize value
I2dsquare=0.0    # initialize to allow for calculation of variance
for n in xrange(numpoints):
    x=numpy.random.uniform()
    y=4.0*numpy.random.uniform()
    I2d=I2d+x**x+2.0*y*y
    I2dsquare=I2dsquare+(x**x+2.0*y*y)**2

# we scale the integral by the total area and divide by the number of points used
I2d=I2d/numpoints
I2dsquare=I2dsquare/numpoints
EstimError=4.0*numpy.sqrt((I2dsquare - I2d**2)/numpoints)  # estimated error
I2d=I2d*4.0
print "Value: %f" %I2d
print "Error estimate: %f" %EstimError

Listing 1: A Python program which demonstrates how to use the Monte Carlo method to calculate the volume below \( z = x^2 + 2y^2 \), with \((x, y) \in (0, 1) \times (0, 4)\).
Sample Results of Monte Carlo Program

<table>
<thead>
<tr>
<th>N</th>
<th>Value</th>
<th>Error Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>37.09</td>
<td>+/- 10.89</td>
</tr>
<tr>
<td>256</td>
<td>44.49</td>
<td>+/- 2.40</td>
</tr>
<tr>
<td>4096</td>
<td>44.50</td>
<td>+/- 0.60</td>
</tr>
<tr>
<td>65536</td>
<td>43.80</td>
<td>+/- 0.14</td>
</tr>
<tr>
<td>∞</td>
<td>44</td>
<td>0.0</td>
</tr>
</tbody>
</table>