Ordinary diff. equations

- First order ODE, one boundary condition:

$$\frac{dy}{dx} = f(x, y), \qquad y(x_0) = y_0$$

- Second order ODE

$$\frac{d^2 y}{dx^2} + f(x, y)\frac{dy}{dx} = g(x, y)$$

$$y(x_0) = y_0, \qquad y(x_1) = y_1$$

ODEs contn'd

- 2^{nd} order ODE can be replaced with a system of 1^{st} order ODEs:

$$\frac{du}{dx} = v \cdot g(x)$$

$$\frac{dv}{dx} = h(x, u)$$

- Typical situation in RT involves two-point boundary conditions

$$y(x_0) = y_0, \qquad y(x_1) = y_1$$

- Alternative is an initial condition

$$y(x_0) = y_0, \qquad \left.\frac{dy}{dx}\right|_{x_0} = y_0^{'}$$

Runge-Kutta

- For a 1st order ODE the Euler method gives:

$$y_{i+1} = y_i + (x_{i+1} - x_i) f(x_i, y_i)$$

this is also first order RK scheme

- Second order RK:

$$k_1 = h \cdot f(x_i, y_i)$$
$$k_2 = h \cdot f(x_i + 0.5h,\ y_i + 0.5k_1)$$
$$k_3 = h \cdot f(x_i + 0.5h,\ y_i + 0.5k_2)$$
$$k_4 = h \cdot f(x_i + h,\ y_i + k_3)$$
$$y_{i+1} = y_i + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5)$$

- Note that RK is directly applicable to a system of ODE and therefore to any order ODE with *initial* boundary conditions

Finite-difference or relaxation method

- For the 1st order ODE:

$$\frac{dy}{dx} = f(x, y)$$

$$y_{k+1} - y_k = (x_{k+1} - x_k) f\left(\frac{x_{k+1} + x_k}{2}, \frac{y_{k+1} + y_k}{2}\right)$$

- For the 2nd order ODE:

$$\frac{d^2 y}{dx^2} = f(x, y)$$

$$\frac{\left(\dfrac{y_{k+1} - y_k}{x_{k+1} - x_k}\right) - \left(\dfrac{y_k - y_{k-1}}{x_k - x_{k-1}}\right)}{\left(\dfrac{x_{k+1} + x_k}{2}\right) - \left(\dfrac{x_k + x_{k-1}}{2}\right)} = f(x_k, y_k)$$

2nd order scheme (cont'd)

$$A_k y_{k+1} + B_k y_k + C_k y_{k-1} = D_k \quad \text{for} \quad k = 2, N-1$$

where

$$A_k = \frac{2}{(x_{k+1} - x_k)(x_{k+1} - x_{k-1})}$$

$$C_k = \frac{2}{(x_k - x_{k-1})(x_{k+1} - x_{k-1})}$$

$$B_k = -\left(A_k + C_k\right)$$

$$D_k = f(x_k, y_k)$$

For k =1 and k =N equations are given by boundary conditions.

- This particular form of 2nd order equation results in 3-diagonal matrix. If $f(x,y)$ has a form $g(x) + \alpha \cdot y$ it is easy to solve:

$$\begin{pmatrix} b_1 & c_1 & & \\ a_2 & b_2 & c_2 & \\ & & \ddots & c_{N-1} \\ & & a_N & b_N \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_N \end{pmatrix}$$

Note also that for $1 < i < \mathrm{N}$: $-b_i = a_i + c_i + \alpha$

- Solving a system of ODEs leads to a block-diagonal SLE!

RK versus Finite-diff.

+ RK typically has a better convergence on a course grid, better scheme and round-off stability (because errors are controlled on every step).
- High accuracy my be very expensive, specially in multidimensions (fast increase of the number of steps).
+ FD converges (if it does) to much more accurate solution. More complex schemes (2nd order) may gain stability and gain convergence speed.
- Very difficult to check the accuracy.

Cook-book: To study the solution locally when you have time use RK. When high accuracy must be combine with high performance use FD.

Partial Differential Equations

- PDEs come in three flavors: *hyperbolic*, *parabolic*, and *elliptic*
- A typical example of a hyperbolic equation is the 1D *wave* equation:

$$\frac{\partial^2 u}{\partial t^2} = v^2 \frac{\partial^2 u}{\partial x^2}$$

where v is the velocity of wave propagation

PDEs examples

An example of parabolic equation is the *diffusion* equation:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D \frac{\partial u}{\partial x} \right)$$

where D (>0) is the diffusion coefficient

PDEs examples

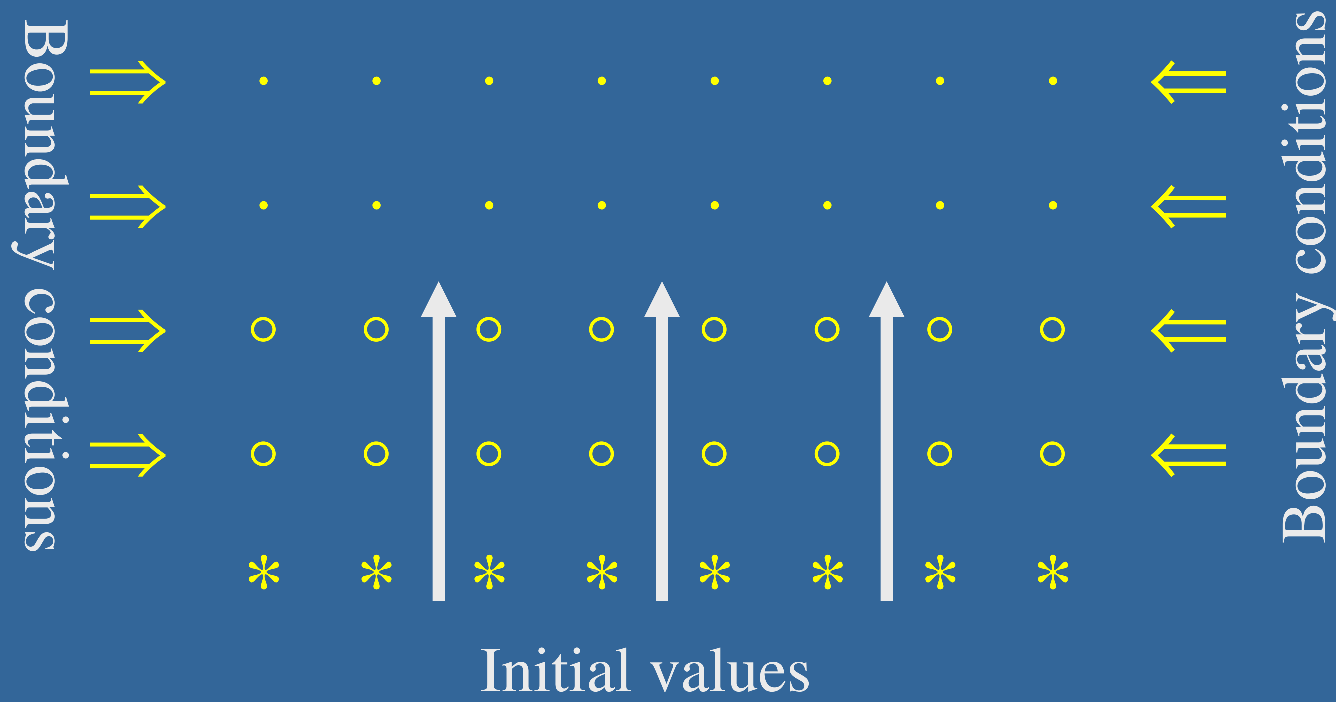
Poisson equation is an example of elliptic type:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \rho(x, y)$$

where ρ is the source function (density of charges). If ρ is zero for the whole domain we get a special case of Laplace equation

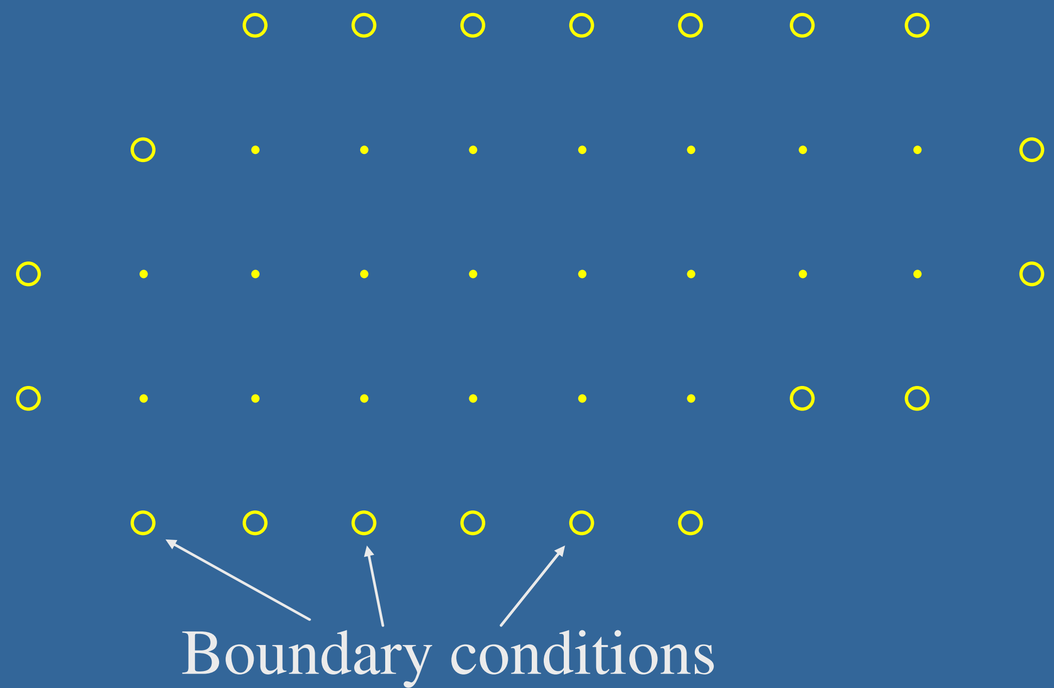
Numerical schemes

Numerically, the first two types are *initial value* problems (Cauchy problems):



Numerical schemes

An elliptic equation is a boundary condition problem:



Numerical scheme for Poisson equation

We assume an equispaced grid with stepsize Δ:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta^2} = \rho_{i,j}$$

In matrix form the scheme looks like this:

$$u_{k+N_x+1} + u_{k-(N_x+1)} + u_{k+1} + u_{k-1} - 4u_k = \Delta^2 \rho_k$$

which is a 5-diagonal SLE

Boundaries and generalization

- The scheme on the previous slide only holds inside the domain. The points for $i = 1, \mathrm{N}_x$ and $j = 1, \mathrm{N}_y$ are described by the boundary conditions leading to a system of linear equation $\mathbf{A}u = b$.
- In our case, elements on each diagonal of A are constant. In general case, matrix elements along diagonals change.
- In 3D more diagonals are present.

Convergence and stability

- Convergence - the property of the numerical scheme to get closer to the exact solution when the grid becomes denser in some regular way.
- Stability - the stability of numerical scheme characterizes the way errors (e.g. finite difference approximation of derivatives) are accumulated during the integration. Stability of the computer implementation of numerical scheme are the way round-off errors are accumulated.

Computational errors

- Floating point numbers are stored in 32- or 64-bit long words:

```
01101000011011011110111001000011
```

sign exponent mantissa

```
1100000011010000101110100100010001011111011100...
```

sign exponent mantissa

- Multiplications/divisions do not loose much precision but subtraction/addition is a danger

Examples on convergence

- Numerical differentiating:

$$\frac{dy}{dx} \approx \frac{y_{i+1} - y_i}{x_{i+1} - x_i}; \qquad \lim_{x_{i+1} \to x_i} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = ?$$

- What causes errors to the solution of SLE? Mathematically $\mathbf{A}x = b$ is ill-defined when $\det \mathbf{A} = 0$. What happens when we solve SLE on a computer?

Diffusion equation

Initial value problem:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D \frac{\partial u}{\partial x} \right)$$

can be approximated as:

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} = D\left(\alpha \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{\Delta x^2} + \right.$$

$$\left. + \beta \frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{\Delta x^2} \right)$$

Stability analysis of numerical schemes

- We represent our solution u_l^m with Fourier components:

$$u_l^m = \xi^m \sum_k v_k e^{ikl\Delta x}$$

- Substituting individual components to the finite difference scheme we find that those with $\left|\xi^m v_k\right| > 1$ are unstable, that is their amplitude grows exponentially with time

Explicit or implicit?

- New sense of stability: *how dramatic will be the solution after many time steps if we change the initial conditions a little bit?*
- Explicit schemes (α=1, β=0) tend to have better convergence
- Implicit schemes (α=0, β=1) tend to be more stable
- Combining the two (α>0, β>0) helps to get the optimal scheme

Integration: Gauss quadratures

- For any “reasonable” function *f(x)* the integral with kernel *K(x)* can be approximated as a sum of function values in nodes x_i multiplied by weights.

$$\int_b^a K(x)f(x)dx \approx \sum_{i=1}^{\mathrm{N}} \omega_i f(x_i)$$

- If f(x) can be represented with orthogonal polynomials, the quadrature formula is exact.

Non-linear equations

For system of non-linear equations we often use Newton-Raphson scheme:

$$F_i(x_1, x_2, \ldots, x_N) = 0, \quad i = 1, \ldots N$$

$$0 = F_i(\vec{x} + \delta\vec{x}) = F_i(\vec{x}) + \sum_{j=1}^{N} \frac{\partial F_i(\vec{x})}{\partial x_j} \delta x_j$$

$$\left(\frac{\partial F_i(\vec{x})}{\partial x_j} \right) \cdot (\delta\vec{x}) = -\left(\vec{F}(\vec{x}) \right); \; \vec{x}_{\text{new}} = \vec{x}_{\text{old}} + \delta\vec{x}$$

Home work

- Write a simple program for 4x4 matrix inversion with Gauss-Jordan elimination and partial pivoting. Test round-off stability by doing 1 million inversions of a 4×4 matrix.

$$\mathbf{A} = \begin{pmatrix} 0.3250 & 0.1030 & 0.1373 & 0.5584 \\ 0.1030 & 0.2034 & 0.5463 & 0.5990 \\ -0.1373 & -0.5463 & 0.9292 & 0.1008 \\ 0.5584 & 0.5990 & -0.1208 & 0.3249 \end{pmatrix} + 10^5$$

In IDL you can do: a=randomu(seed,4,4)+1.e5

- Propose the scheme (flow chart) for solving SLE with 4x4 block-diagonal matrix
- Write (or use NR routine) 4th order RK and FD scheme for the equation:

$$\frac{d^2 y}{dx^2} = e^x + y; \quad y(0) = 1; \quad y(50) = 10^3$$