Miniproject

Warwick RSE

Every project is an opportunity to learn, to figure out problems and challenges, to invent and reinvent.

David Rockwell, American Architect and Designer

Project

- Project isn't much harder than the existing assignments
  - Working in a group is *part of the difficulty*
  - It is not easy to program in a group and make everything work together
- Problem is a simple electrostatic problem
  - Calculate the electric field due to a distribution of charge in 2D and the motion of a single electron in response to that field
  - All done in double precision

Project

- This is not an algorithms course
- The algorithms in this mini project are a bit more complex that those in the assignments
- We will give you quite explicit versions of the algorithms that you just have to change into Fortran code
- If you find that you are having problems with the algorithms **do ask!**
  - We are not testing you on algorithms!

Project

- The physics of this problem are
  - Imagine that you have a stable distribution of charges that don’t change in time
    - Think very heavy ions
  - From this you can get charge density, from which you can get the electric fields that are also constant in time
  - You then move a single electron in the field from the ions
    - The ions will move eventually but not on the timescale of the electron
    - Classical scale separation

Project

- We are going to provide you with some code elements that implements a handful of slightly messy but important bits

- Mostly you are going to have to design the structure of the code yourselves

- This code only **needs** you to use allocatable arrays, functions, modules and kinds

    - There is scope for types as well but make sure that everyone in your group is happy with everything

Equation for Field

$$\nabla.\mathbf{E} = \frac{\rho}{\epsilon_0}$$

$$\mathbf{E} = \nabla\phi + \nabla \wedge \mathbf{A}$$

Helmholtz decomposition

$$\nabla.\nabla \wedge \mathbf{F} = 0$$

$$\nabla.\mathbf{E} = \nabla^2\phi = \frac{\rho}{\epsilon_0}$$

Equation for Field

$$f(x+dx,y) = f(x,y) + dx\frac{\partial f}{\partial x}|_{x,y} + \frac{dx^2}{2}\frac{\partial^2 f}{\partial x^2}|_{x,y} + O(dx^3)$$

+

$$f(x-dx,y) = f(x,y) - dx\frac{\partial f}{\partial x}|_{x,y} + \frac{dx^2}{2}\frac{\partial^2 f}{\partial x^2}|_{x,y} + O(dx^3)$$

=

$$f(x+dx,y) + f(x-dx,y) = 2f(x,y) + dx^2\frac{\partial^2 f}{\partial x^2}|_{x,y}$$

Equation for Field

$$f(x+dx,y) = f(x,y) + dx\frac{\partial f}{\partial x}|_{x,y} + \frac{dx^2}{2}\frac{\partial^2 f}{\partial x^2}|_{x,y} + O(dx^3)$$

+

$$f(x-dx,y) = f(x,y) - dx\frac{\partial f}{\partial x}|_{x,y} + \frac{dx^2}{2}\frac{\partial^2 f}{\partial x^2}|_{x,y} + O(dx^3)$$

=

$$\frac{\partial^2 f}{\partial x^2}|_{x,y} \approx \frac{f(x+dx,y) - 2f(x,y) + f(x-dx,y)}{dx^2} + O(dx^3)$$

Equation for Field

$$f(x+dx,y) = f(x,y) + dx\frac{\partial f}{\partial x}|_{x,y} + \frac{dx^2}{2}\frac{\partial^2 f}{\partial x^2}|_{x,y} + O(dx^3)$$

-

$$f(x-dx,y) = f(x,y) - dx\frac{\partial f}{\partial x}|_{x,y} + \frac{dx^2}{2}\frac{\partial^2 f}{\partial x^2}|_{x,y} + O(dx^3)$$

=

$$\frac{\partial f}{\partial x}|_{x,y} \approx \frac{f(x+dx,y) - f(x-dx,y)}{2dx} + O(dx^3)$$

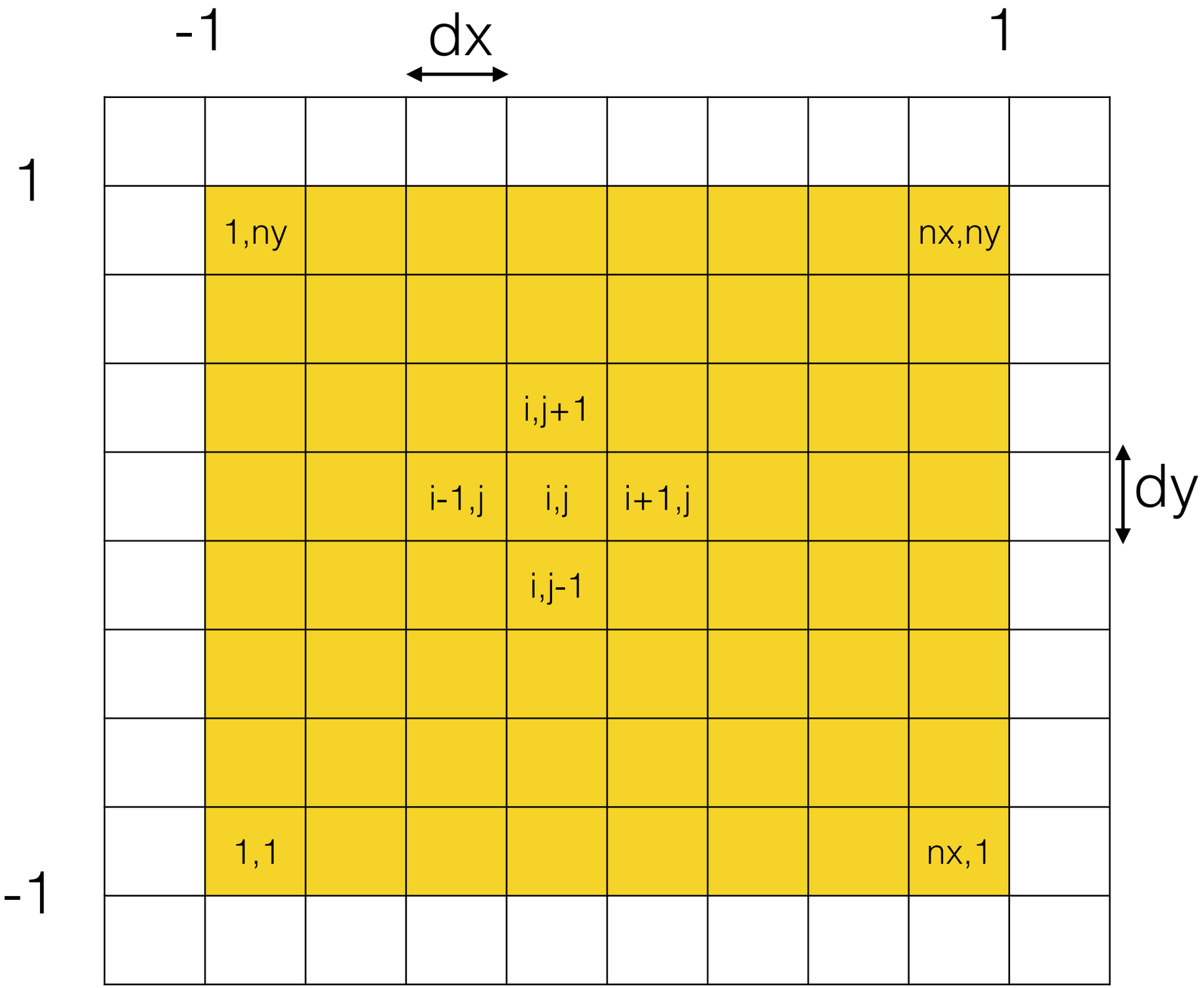
Equation for Phi

$$\nabla^2 \phi = \frac{\rho}{\epsilon_0}$$

$$\frac{\phi(i-1,j) - 2\phi(i,j) + \phi(i+1,j)}{dx^2} + \frac{\phi(i,j-1) - 2\phi(i,j) + \phi(i,j+1)}{dy^2} = \frac{\rho(i,j)}{\epsilon_0}$$

- Solve on a grid for phi at each grid point
- Then can construct the electric field from that when needed
- Without loss of generality, assume epsilon$_0$ is 1

Grid



Grid

- It is quite important that you set up your grid correctly otherwise you will break symmetry which can change answers by quite a bit

- -1 is at the left hand edge of the first cell
  +1 is at the right hand edge of the last cell
  Everything else is defined at the centre of the cells

- This means that there is an “extra” cell length across the whole domain because of that extra half cell at the start and end

- Provided code will get this right

Solving field Equation

- Can’t algebraically rearrange equation for phi at a single (i,j)
    - Have to solve simultaneously as a matrix equation for all (i,j)
- Here we’re going to use an iterative technique called Gauss-Seidel
- Iterative as in you keep applying the same rule until you have a good enough approximation to the true answer

Solving field Equation

$$\phi(i,j) = -\frac{\rho(i,j) - \frac{\phi(i+1,j)+\phi(i-1,j)}{dx^2} - \frac{\phi(i,j+1)+\phi(i,j-1)}{dy^2}}{\frac{2}{dx^2} + \frac{2}{dy^2}}$$

- Important to note that you update phi **as you go**
  - You **DON'T** update it to a temporary array
  - If you do then you're using a related method called the Jacobi method
  - Jacobi converges slower than Gauss-Seidel

Solving field Equation

$$\frac{\phi(i-1,j)-2\phi(i,j)+\phi(i+1,j)}{dx^2}+\frac{\phi(i,j-1)-2\phi(i,j)+\phi(i,j+1)}{dy^2}-\rho=\text{error}$$

- Next thing you need is a condition for when to stop iterating
  - Lots of choices!
- Calculate the error in the current iteration using the above formula at each point and sum the absolute values over the domain
- Find the ratio of this error to the RMS value of **just the numerical derivative**
- When this reaches 1 x 10^{-5} you have a good enough answer for this problem

Solving field Equation

$$e_{tot} = \Sigma\mathrm{ABS}(\frac{\phi(i-1,j) - 2\phi(i,j) + \phi(i+1,j)}{dx^2} + \frac{\phi(i,j-1) - 2\phi(i,j) + \phi(i,j+1)}{dy^2} - \rho)$$

$$d_{rms} = \sqrt{\Sigma(\frac{\phi(i-1,j) - 2\phi(i,j) + \phi(i+1,j)}{dx^2} + \frac{\phi(i,j-1) - 2\phi(i,j) + \phi(i,j+1)}{dy^2})^2}$$

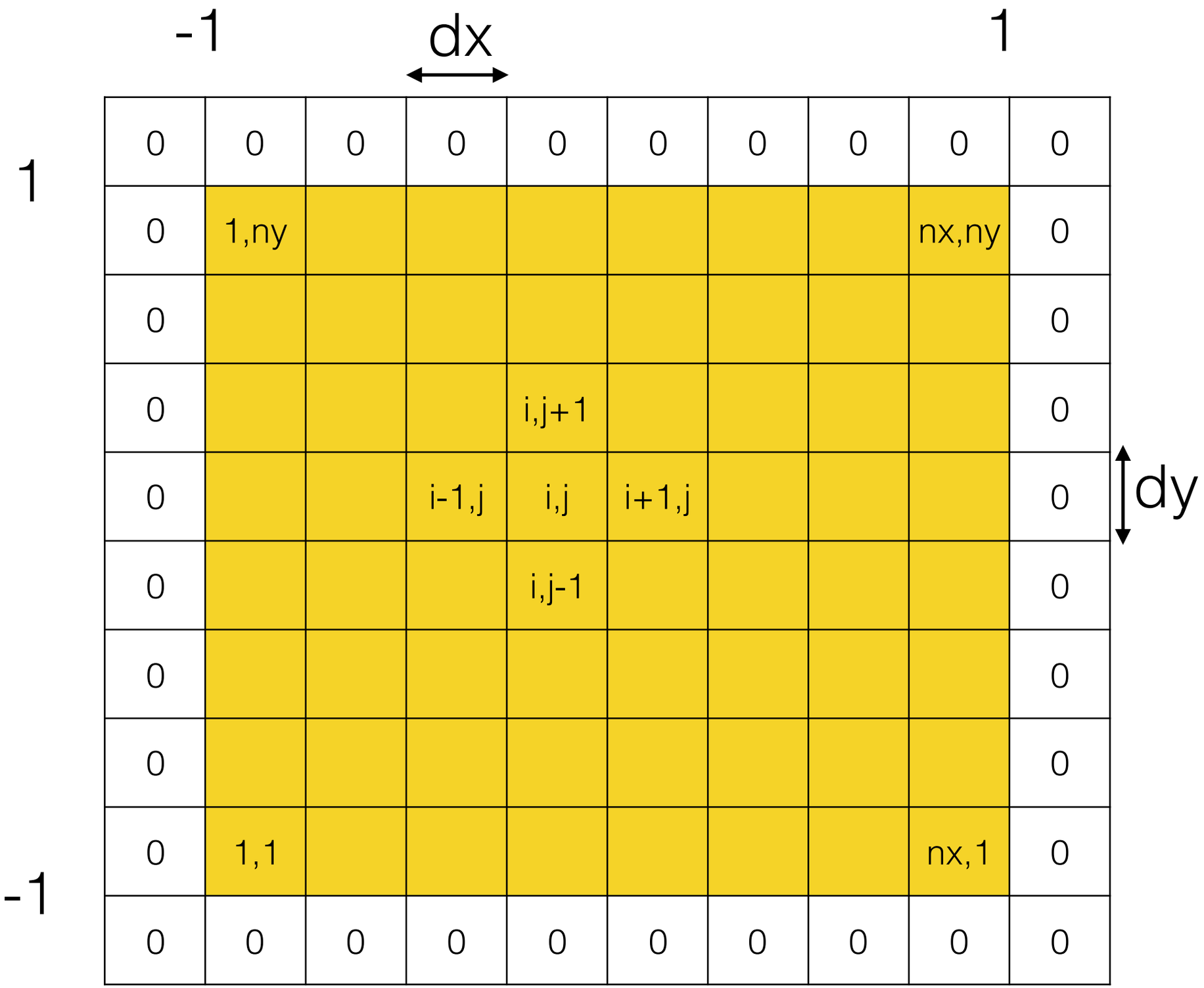
Iterate Until $\frac{e_{tot}}{d_{rms}} \leq 10^{-5}$

HINT : can d_{rms} be zero?

Solving field Equation

- Final thing that you need are boundary conditions
- Implement these as **ghost cells**, an extra strip of cells along each edge of the domain
- Recommend but not required that you use the explicit array bounds to have arrays that run (0:nx+1, 0:ny+1) so that the “real” domain runs 1->nx, 1-> ny
- For this simple problem just set these ghost cells to 0 and never modify them

Grid


-1
dx
1
1
0 0 0 0 0 0 0 0 0 0
0 1,ny nx,ny 0
0 0
0 i,j+1 0
0 i-1,j i,j i+1,j 0
0 i,j-1 0
0 0
0 0
0 1,1 nx,1 0
-1
0 0 0 0 0 0 0 0 0 0
dy

Moving a particle in the field

$$E_x(i,j) = \frac{\phi(i+1,j) - \phi(i-1,j)}{2dx}$$

$$E_y(i,j) = \frac{\phi(i,j+1) - \phi(i,j-1)}{2dy}$$

- Create two new arrays for Ex and Ey derived from phi
  - These should only be from 1->nx and 1->ny, you need to use the ghost cells on phi to calculate E but you only need the real cells for E itself
- Now you can use the Lorentz force law to calculate the force due to your original charge density on a test particle

Velocity Verlet Integration

$$a^0 = \frac{q\mathbf{E}(\mathbf{x}^0)}{m}$$

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \mathbf{v}^n dt + \frac{1}{2}(\mathbf{a}^n)^2 dt^2$$

$$\mathbf{a}^{n+1} = \frac{q\mathbf{E}(\mathbf{x}^{n+1})}{m}$$

$$\mathbf{v}^{n+1} = \mathbf{v}^n + dt\frac{\mathbf{a}^{n+1} + \mathbf{a}^n}{2}$$

- Start by specifying an initial position (**x**) and velocity (**v**) for your test particle
- Calculate the initial acceleration (**a**) from the Lorentz force law
- Similar simplification, assume that m =1 and q = -1
- Then follow the rule on the left to advance x, a and v
- If the particle's position moves outside the grid that you have E on then stop iterating

E field at particle position

- You know your E field values at fixed positions in the domain

- Your particle is moving freely in space and isn't locked to your grid

- You should **really** interpolate from the surrounding grid points to get the field at the particle position but that is quite hard to do right

- For this, simply use the field in the cell that the particle is in

E field at particle position

- Fortunately for a simple rectangular grid like this don't have to do anything clever to find out what cell you are in

- cell_x = FLOOR((part_x - 1.0_dp)/dx) + 1
  cell_y = FLOOR((part_y - 1.0_dp)/dy) + 1

- This is working out how many cells of width dx or dy you are from the bottom of the domain (-1)

- Then add one because your array starts from 1 in each direction

Output

- Your code should output the following as a NetCDF file
  - The charge density rho (1:nx, 1:ny)
  - The potential phi (1:nx, 1:ny)
  - Ex and Ey (1:nx, 1:ny)
  - For your particle
    - Position at every iteration
    - Velocity at every iteration
    - Acceleration at every iteration

Control of your code

- Your code should be able to take the following parameters as command line arguments (you can use our routines for parsing the arguments if you wish but remember to give credit)
  - nx - number of grid points in x
  - ny - number of grid points in y
  - problem - string saying which test problem to run

Test problems

- “null” - rho = 0 everywhere, particle starts at 0,0. Initial particle velocity (vx=0.1,vy = 0.1)

- “single” - rho is of the form EXP(-(x/0.1)^2 - (y/0.1)^2). Particle starts at 0.1, 0.0. Zero initial velocity

- “double” - rho is of the form EXP(-((x+0.25)/0.1)^2 - ((y+0.25)/0.1)^2) + EXP(-((x-0.75)/0.2)^2 - ((y-0.75)/0.2)^2). Particle starts at 0.0,0.5. Zero initial velocity

- All problems should solve to find the E field for the specified charge density rho first and then move the particle in that field

Test problems

- Should move the particle for 1000 iterations with dt =0.01

- There are proper ways of working out that timestep but they aren't really instructive here

- 0.01 will work pretty well

- Remember epsilon0 =1, m=1, q=-1

    - If you put in SI units then 0.01 isn't close

Control script

- You should provide a bash script that builds your code, runs your code on a 100x100 grid with the “single” test problem and then starts a visualisation in Python. The visualisation should plot

  - Ex as a pseudo colour plot

  - The particle X position vs the particle Y position as a scatter plot

- If you wish to use Make or another tool to build your code you may do so but this must be called from your shell script

- There is no need to have your script test for libraries etc. you can assume that everything needed is present

Suggestions

Splitting up the work

- You are working in groups here. The work naturally splits up into
  - Gauss-Seidel solver
  - Particle mover
  - NetCDF output
  - NetCDF input and visualisation

Splitting up the work

- In pairs
  - Each person does one numerical solver and one part of the IO system
- In threes
  - One person does Gauss-Seidel, one the particle mover and one the IO system

Designing the Structure

- You will have to work out between yourselves how you want the bits of the solver to fit together
- You will need arrays holding rho, phi, Ex and Ey somewhere
  - They can be module variables
  - They can be in types

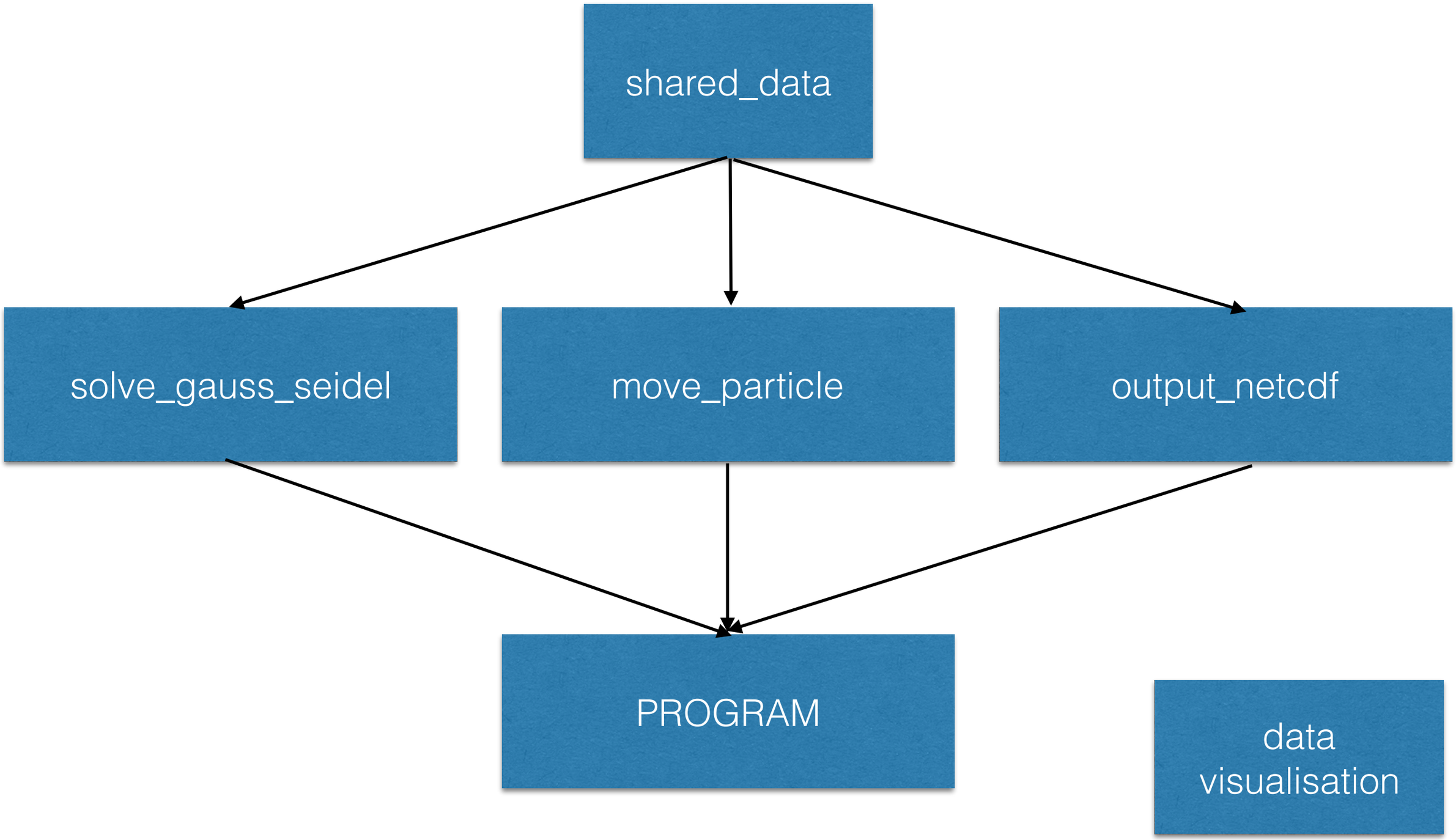
Designing the Structure

- You will need arrays holding the positions, velocities and accelerations of your particle (remember that we want the full time histories of the particle so you have to store everything for all time)

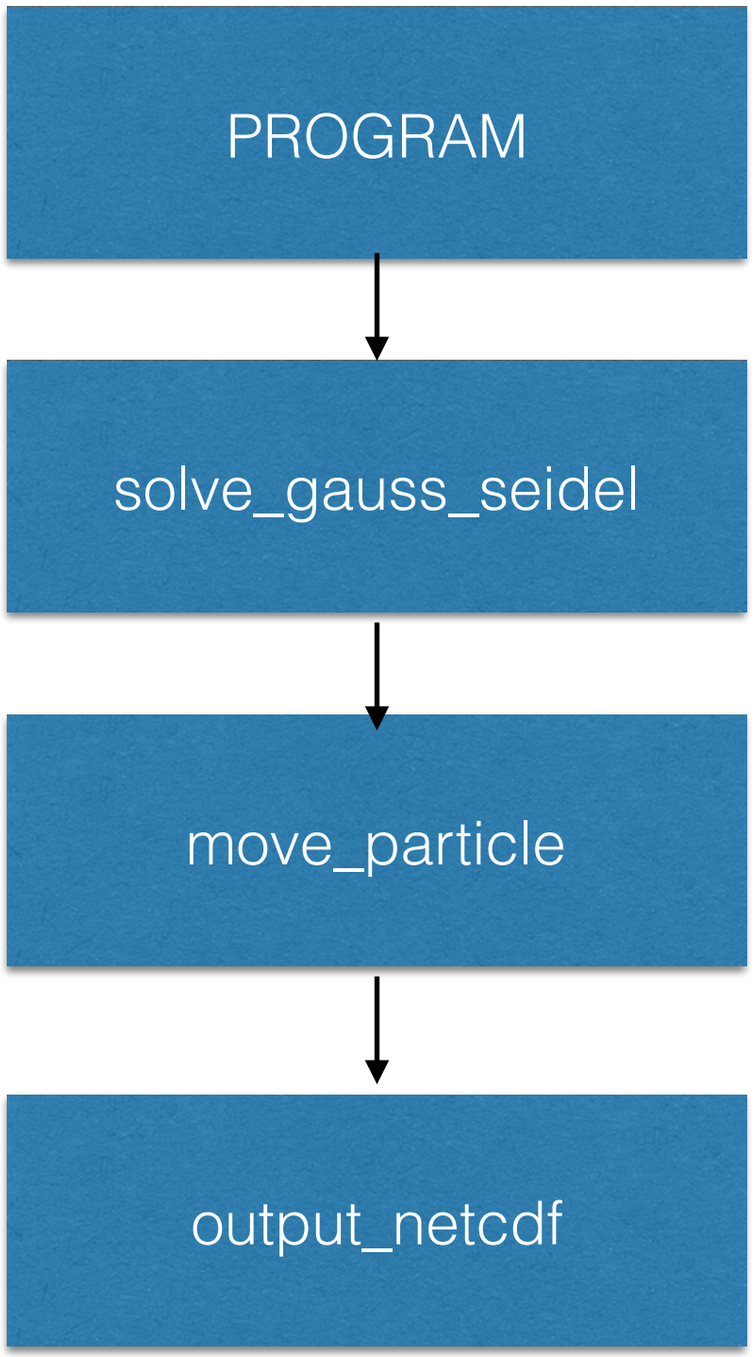
- They can be module variables again **but** for this one I'd recommend using types

- An obvious extension of the work is to have more than one particle moving in the field and this is much simpler if you have a particle type

Dependencies



Calling Order



Designing the Structure

- Before starting we would suggest starting with pen and paper
  - Agree on data structures
  - Agree on variable names
  - Agree on function interfaces
- Once that is done get started on roughing out the code
  - We'd suggest using github

Designing the Structure

- One person creates a github repository
    - Invite other people as collaborators
- Everyone works on a different file
    - Minimises the risk of merge conflicts

Testing your results

- Remember a few things
  - Potential will be peaked somewhere near the peak in the charge density
  - To a good approximation you can replace the distribution of charge with a single point charge at the peak of the distribution
  - An electron would be attracted to that point charge
  - Your electric field should have the same property
  - Note that your charge density has symmetry. You would expect the potential to have similar symmetry

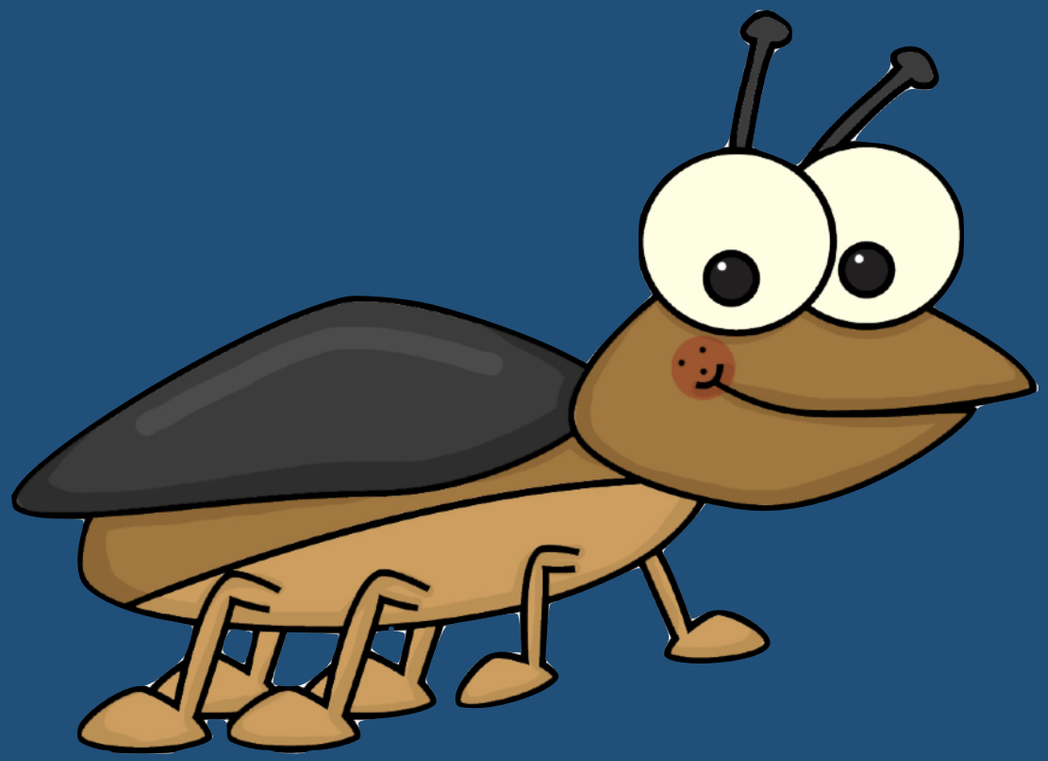
Marking

How we're marking

- What aren't we marking?
  - Use of clever or advanced language features
  - Correctness of solution (within limits)
    - A few minor mistakes that leads to you getting sensible answers that are different to ours isn't going to factor into marking
    - Results that are obviously wrong are!
      - Being able to code up an algorithm is a part of the skill set
    - Performance

How we're marking

- What **are** we marking?
  - Conformance to standards
    - Including memory leaks and array bounds errors
  - Comments and readability
  - Getting the right general answer
    - (Mostly) correct implementations of the algorithm
  - Evidence of working together



The End