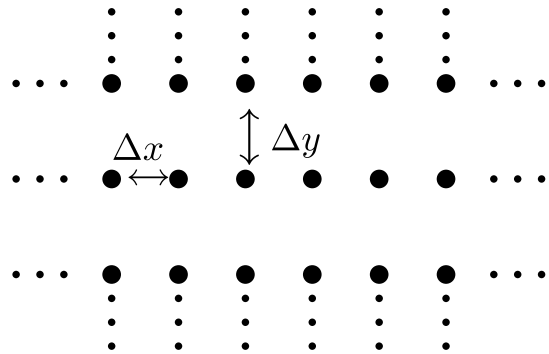
Parabolic equations in 2 spatial dimensions *(animations)*

$$\text{Heat equation: } \partial_t T = \nabla \cdot (\kappa \nabla T)$$

$$\text{Constant diffusivity: } \partial_t T = \kappa \nabla^2 T$$

$$\text{2-D: } \partial_t T = \kappa(\partial_x^2 T + \partial_y^2 T)$$

Consider Cartesian, equally-spaced grid



A. Label points, (i,j). This is called a structured grid because i,j ↔ x,y is 2-D to 2-D mapping and i+1,j is next to i,j in physical space. Unstructured is i ↔ x,y: 1-D to 2-D: nodes and cells

B. *Euler Explicit*

In explicit case, 3D not much harder than 2-D

$$\frac{T_{i,j}^{n+1} - T_{i,j}^n}{\Delta t} = \kappa(\delta_x^2 T^n + \delta_y^2 T^n)$$

Generally, march in time $T^{n+1} = T^n + RHS^*\Delta t$. Discretize RHS by 2nd order central:

$$\delta_x^2 T + \delta_y^2 T = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2}$$

or, with

$$\alpha_x = \kappa \Delta t/\Delta x^2, \alpha_y = \kappa \Delta t / \Delta y^2$$

$$T_{i,j}^{n+1} = T_{i,j}^n + \alpha_x(T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n) + \alpha_y(T_{i,j-1}^n - 2T_{i,j}^n + T_{i,j-1}^n)$$

Coeff of $T_{i,j}$ is $1-2\alpha_x - 2\alpha_y$: Good idea for $1-2\alpha_x-2\alpha_y > 0$; i.e. $\alpha_x + \alpha_y < 1/2$.

If $\Delta x=\Delta y$, this constraint becomes $\alpha_x < 1/4$ or $\Delta t < \Delta x^2/4\kappa$

This is 1/2 of 2-D time-step. Same result can be found by VonNeuman stability analysis

Pseudo code

Initialize T(x,y) including boundaries. For fixed T boundary just update interior:

```
DO k=2,N-1
   DO j=2,N-1
      ΔT(j,k) = αx(T(j+1,k) -2T(j,k) +T(j-1,k)) + αy(T(j,k+1) -2T(j,k) +T(j,k-1))
   ENDDO
ENDDO

T(2:N-1,2:N-1) = T(2:N-1,2:N-1) + ΔT(2:N-1,2:N-1)
↑                ↑                ↑
T(t+Δt)          T(t )            ΔT
```

What to do about gradient b.c.? Go around boundary updating.

E.g., T(1, 2:N-1) = T(2, 2:N-1) - Qw Δx/κ

is first order in Δx, heat flux boundary condition.

C. Easy to increase accuracy in time: semi-discrete

$$\frac{dT_{j,k}}{dt} = \frac{\kappa}{\Delta x^2}(T_{j+1,k}+T_{j-1,k}-2T_{j,k})+\frac{\kappa}{\Delta y^2}(T_{j,k+1}+T_{j,k-1}-2Tj,k) = RHS_{j,k}$$

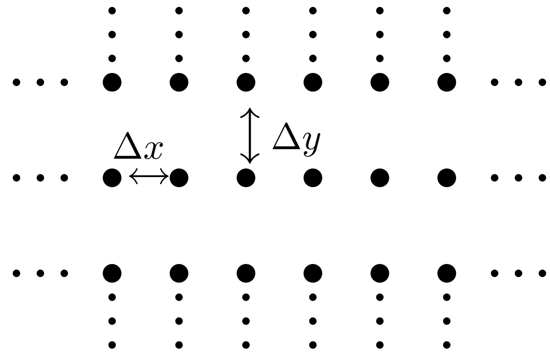
Use R-K or A-B on system of o.d.e.s

Schematic of RK solution to 2-D heat equation

```
T(:,:)=T0(:,:)      ! Initial conditions; assume constant T b.c.
DO t = 0,Tend, Δt
                    !  Tn in Tn+1 out
   call rk2(II,JJ,RHS,T, Δt)  ! advance from t to t+ Δt
! Plot at time = t, or other processing,  as needed
ENDDO
!---------
! Right side used by nth-order RK solver
  SUBROUTINE RHS(II,JJ,Tp,T,Δt)
!---------
   REAL(8) :: T(II,JJ),Tp(II,JJ)
   TP = 0.
   DO j=2,JJ-1
      DO i=2,II-1
         Tp(i,j) = (T(i+1,j)-2T(i,j) +T(i-1,j))/Δx2 + (T(i,j+1)-2T(i,j) +T(i,j-1))/Δy2
      ENDDO
   ENDDO
   RETURN
```

D. Matrix-vector viewpoint

(i,j) ↔ (j-1)*I + i



1. Data are stored in 2-D array T(i,j). Number of locations = I*J. Let's think of this as a 1-D solution vector.
2. Label with a single index. Count grid points:

   j=1: 1, 2, 3... I
   j=2: I+1, I+2, I+3... 2I
   j=3: 2I+1,2I+2, 2I+3... 3I
   .....
   j=J: (J-1)*I+1, (J-1)*I+2, (J-1)*I+3... I*J

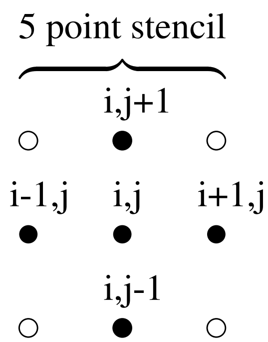
   I.e. (i,j) ↔ (j-1)*I + i (Count on grid from LL to UR)

   T[i,j] ↔ T[(j-1)I + i]

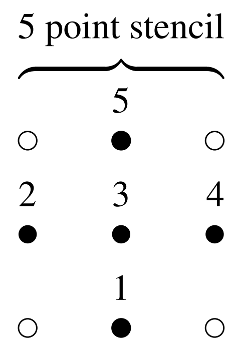
   Why talk about single numerical ordering? To understand matrix (next lecture).

B. Terminology: finite difference formula involves $T_{i+1,j}, T_{i,j}, T_{i-1,j}, T_{i,j+1}, T_{i,j-1}$:
called 5-point stencil

$$\delta_x^2 T + \delta_y^2 T = \frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2}$$



or



Count from i,j to i,j+1: i+1 to I = I-i points along j[th] row
then i points along j+1[th] row, gives I memory locations
between adjacent points.

Same if you count from i,j-1 to i-1,j. Adding the point at i,j gives 2 I+1 points inside stencil
defines *Bandwidth*